Geek 1 is about basics of programming where we have tried to answer the following questions :

Contents

[hide]

- 1 What is a computer program?
- 2 Why do we need to program ?
- 3 Can anybody learn programming?
- 4 Are there any prerequisites for the course?
- 5 Where to start?
- 6 What is programming?
- 7 How to give instructions?
- 8 Introduction to Scratch

What is a computer program?[edit]

A computer program is a well defined set of instructions that is used to make computer to do an expected task and give a desired output,written in any computer programming language like c,c++, Java, Python etc.It acts as an interface between the user's ideas and the system.

Why do we need to program ?[edit]

Today in the digital world, most of what you see on your cell phone, computers or be it on the internet, webpage, etc., all of these are some programs written by someone. Someone should have written a piece of code for all these things to work on our fingertips. It makes our life really simple. Programming is about getting things done really fast. So, in order to develop or understand the functioning of this digital world, and make our life little faster and easier, we must learn to program.

Can anybody learn programming?[edit]

#Top

Yes, Programming is actually quite commonsensical. But there is one problem that a student faces majorly- a whole lot of discouragement. So, its like if you come and tell me you want to learn French and I start off giving you literature in French, then you will be obviously discouraged. The next discouraging factor for someone to learn programming is that they end up feeling extremely discouraged by looking at someone who does really well. First one is whole lot of details we start with some complicated stage of programming we don’t start from scratch, secondly, people around you sort of discourage you and put you off.

A one-word answer to the question is YES, in fact, anyone can program and it's a fairly easy process.

Are there any prerequisites for the course?[edit]

#Top

There are no prerequisites required for the course. We only expect you to know a little bit of computer handling and interest in learning. Also, one should have the ability to think logically.

Even, a 10th Standard/High school passed student can also finish this course.

But, one should have a basic knowledge of using a computer and mouse.

Where to start?[edit]

#Top

Today there are so many languages. People often get confused about where to start. The important part in programming is understanding the programming logic behind every problem, not the programming language. You should start giving instructions to the machine. Knowing and trying to implement the Art of Articulation.

What is programming?[edit]

#Top

Programming is the process of writing computer program which is a sequence of instructions which when given to computing machine accomplishes the desired task in specific pattern as instructed. It is also the process of representing an item as a program and making suitable rules for its successful execution.

To enable the computer to undertake any task, our instructions have to be converted (from English) into a language or code, such as C, C++, Java, Python, which can be understood by the computer.

How to give instructions?[edit]

Instructions are programming language statements written stepwise in a program to do a specific task.

Instructions should be crystal clear. They should be articulated in such a way such that they are understood by each and everyone who are being instructed.

Give instruction --> Follow --> Get the task done

#Top

Introduction to Scratch[edit]

Scratch is a very easy programming language which can be understood by even an 8-year-old. It is a block-based visual programming language and online community. One of the easiest way to learn control structures, without a bit code written.

It can be downloaded from https://scratch.mit.edu/download. Scratch works on Windows and Mac Operating Systems. You can also perform task online by going on this link https://scratch.mit.edu/projects/editor/?tutorial=getStarted.

Week 2

Lists[edit]

The lists are containers that hold some other objects in a given order. It usually puts into practice the sequence protocol and allows programmers to add or remove objects from that sequence. Each element of the sequence is assigned a number, i.e., the index and the first index is 0 (zero). This versatile data-type of Python is written in a sequence of list separated by commas between expressions.

For separating inputs use the command: *a = input().split()* and to access them individually use: a[0], a[1] ....

To print in the same line with space, use the code: *print(num, end="")* Lists

To read more than one value of list in a single line, use: *list1 = list(map(int,input().split()))*

'''CREATING LISTS''':

To build a list, just put some expressions in square brackets. *The syntax is: lst1 = [ ] # lst1 is the name of the list* lst2 = [expression1 , …. , expression_N] **e.g.,** *lst1 = ['computersc', 'IT', 'CSE'];* lst2 = [1993, 2016]; lst3 = [2, 4, 6, "g", "k", "s"];

[[**ACCESSING LIST VALUES:**]] List apply the standard interface sequence in which len(L) returns the number of items present in the list and L[i] represents the item at index i. Also L[i:j] returns new list containing objects within 'i' and 'j'.

'''PROGRAM TO EXPLAIN HOW TO ACCESS LISTS:'''

e.g., *lst1 = ['computersc', 'IT', 'CSE'];* lst2 = [1993, 2016]; lst3 = [2, 4, 6, "g", "k", "s"]; **OUTPUT:** *print ("lst1[0]", lst1[0])* print ("lst3[2:4]", lst3[2:4]) lst1[0] computersc lst3[2:4] [6, 'g']

*[[**UPDATING LISTS:**]]* **e.g.,** Program to show how to add/update single or multiple elements in a list:

lst1 = ['computersc', 'IT', 'CSE'];

print ("Second value of the list is:"); print (lst1[1])

lst1[1] = 'Robotics' print ("Updated value in the second index of list is:");Link title print (lst1[1])

OUTPUT: Second value of the list is: IT Updated value in the second index of list is: Robotics

'''DELETING ELEMENTS FROM LISTS:'''

To remove an element from the list, we can use the del-statement. The syntax for deleting an element from a list is:

SYNTAX: *del list_name[index_val];*

Deletion of elements could also be done by using remove(). Syntax:

```
list_name.remove(element)
```

example, let's take a list,

```
li = [34,67,12,90]
```

to remove a specific element, let's say 12, all you have to do is, type

li.remove(12)

the new list would be, [34,67,90]

pop()

```
pop is another way of removing elements from the list.
It allows us to delete the last element from the list, just the way it does with a stack structure.
It also returns the popped element
```

Syntax

```
list_name.pop()
```

example

```
li = [12,34,56,24]
a=li.pop()   # returns 24
print(a) #prints 24
```

And the new list would be, [12,34,56]

List Functions[edit]

"all" & "any" take a list as an argument, and return True if all or any(respectively) of their arguments evaluate to True(& False otherwise).

The function "enumerate" can be used to iterate through the values & indices of a list simultaneously.

Introduction to Python[edit]

Python is a widely used programming language which allows programmers to write the program in fewer lines of code. It was designed by *Guido van Rossum* in 1991. Python uses an interpreter instead of compiler. Basically, the main difference between compiler and interpreter is that the compiler takes the code as 'whole' and compiles it. for example, in *C* language and on the other side in interpreter. For example: languages like Python. It interprets the written code 'line by line', and if somewhere error is found. It stops interpreting the code from there itself. Here, in our course we will be using *Spyder IDE*. Spyder is an integrated development environment with editing options

to assist the programmers in writing programs. Spyder IDE is divided into two panes, one is console and other one is where program is written.

Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc). Python has a simple syntax similar to the English language. Python has syntax that allows developers to write programs with fewer lines than some other programming languages. Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.

Python can be treated in a procedural way, an object-orientated way or a functional way.

Installing Anaconda[edit]

#Top

This article is all about installing and troubleshooting Anaconda and Spyder on different platforms.

Windows[edit]

Step 1 -> Download the installer from here

Download Link. Make sure that you download the version compatible to your operating system architecture. If you have a 64 bit OS then you can download either 64 bit of 32 bit, but if you have 32 bit OS then you should download the 32 bit version only.

Step 2 -> Double click on the .exe file that you downloaded just now and follow the instructions. If you are not sure of the installation options then just choose the default.

Step 3 -> To test your installation, open cmd (go to Start -> then type cmd in the search bar and hit enter). There type **conda list**. You will see a list of packages installed if not, then there was some problem.

Mac[edit]

Step 1 -> Download the installer from here Download Link.

Step 2 -> Double click on the .pkg file that you just downloaded. If you are not sure of the installation options then just choose the default.

Step 3 -> To test your installation, open terminal. There type **conda list**. You will see a list of packages installed if not, then there was some problem.

Linux[edit]

Step 1 -> Download the installer from here Download Link.

Step 2 -> Go to the folder where you downloaded the .sh file and type following command

```
bash Anaconda-latest-Linux-x86_64.sh
```

If you are not sure of the installation options then just choose the default.

Step 3 -> To test your installation, open terminal. There type **conda list**. You will see a list of packages installed if not, then there was some problem.

Troubleshooting Spyder[edit]

#Top

In most of the cases, it happens that Spyder is already running and you have initiated one more instance of it. Please make sure that spyder is not running already. If you are on Linux OS, please launch spyder through terminal. Just open the terminal and type **spyder** and it will launch. When Spyder opens, make sure you see Anaconda with the version written on the Ipython console. If not, try changing the path of your anaconda. Follow these steps according to your OS version.

Windows[edit]
Click Here

Linux[edit]
Click Here

Mac[edit]
Click Here

In most of the cases relaunching Spyder works. A more general Troubleshoot is present Here

How to run a program in Spyder IDE?[edit]

Go to Run on the menu bar. Click Run->Run.

For Loop and its importance[edit]

A loop is an essential part in coding. It gives us the ability to repeat a part of the code we have written already. This can make the code short and effective. for loop is a type of loop Its syntax is:

```
for i in range(10):
    statement
```

i is a variable whose initial value will be 0 ((if initial value is not defined). This i will be incremented by 1 every time the loop runs. So, value of i will be 0,1,2,3,4,5,6,7,8,9

For making a loop of 20 all we need to do is

```
for i in range(20):
    statement
```

Note: Giving a colon " : " and tab space is essential in loop. A tab space insures that the code you are writing is in the loop If we don't insert a tab space the computer will consider that the code is outside the loop.

The for loop is commonly used to repeat some code a certain number of times. This is done by combining for loops with range objects.

>>>for i in range(5):

```
    print("Hello")
```

Will print Hello 5 times

The for loop can also have a starting and end range. We can print all natural numbers from 1 to 10.

```
>>>for i in range(1, 11):

    print(i)
```
Will print 1,2,3,4,5,6,7,8,9,10

Note: The loop executes for i =1 to i=10 and when i = 11, the loop terminate. The end value of range is never counted for execution.

Difference between '=' and '==' in Python[edit]

The '=' operator is an Assignment Operator. So, a = 3 will assign 3 to variable a.

The '==' operator is a Logical Operator. So, a == 3 will check if value of variable a is 3 or not.

Printing Statements in Python[edit]

The print function is used to produce output. It displays a textual representation of something to the screen.

When a string is printed, the quotes around it are not displayed.

For example, **>>> print('print("Hello")')** Will give the output as **print("Hello")**

Input[edit]

To get an input from the user, we use the intuitively named "input" function. The function prompts the user for input and returns what they enter as string(With the contents automatically escaped). **Syntax:** input("Enter text here")

IF ELSE Statements[edit]

Syntax: >>>if(expression):

```
        statements
    else:
        statements
```

Python uses indentation to delimit the blocks of code (It doesn't use curly braces).

If statements can be nested, one inside other. This means **inner if statement** in the statement part of the outer one. This is one way to see whether multiple conditions are satisfied.

We can also use chain of if & else statements to determine which option in a series of possibilities is true.

elif Statements[edit]

The **"elif" (Short for else if)** statement is a shortcut to use when chaining multiple if & else statements. A series of if elif statements can have a final else block, which is called if none of the if or elif conditions is true.

Example :>>> if(a>b): (statement) elif(b>a): (statement) elif(c>b): (statement) else(a>c) (statement)

Range[edit]

The range function creates a sequential list of numbers.

>>>number = list(range(10))

>>>print(number)

Will give the output: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] '

The call to list is necessary because range by itself creates a "range object" & this must be converted to a list if you want to use it as one.

If range is called with one argument, it produces an object with values from 0 to that argument. If it is called with two arguments, it produces values from the first value to the second.

Range can also have a third argument, which determines the interval of the sequence produced. This third argument MUST be an integer.

>>>number=list(range(5, 20, 2))

Will give the output: [5, 7, 9, 11, 13, 15, 17, 19] '

>>>range(n) is implicitly from 0 to n-1.

>>>range(i, j, k) produces a sequences in steps of k. negative k counts down.

Sequence produced by range() is not a list.

Use list(range(...)) to get a list.

Week 3

Lecture Notes:Week 3

From JOCWiki

Contents

[hide]

- 1 Data structures
- 2 General Features of List

- 3 **Week-3 Game programs**
  - 3.1 fizz buzz simple with if conditions
  - 3.2 **Crowd Computing Code**
    - 3.2.1 To plot data using Python
  - 3.3 **Jumbled Words Game**
- 4 Useful Functions & Methods for Lists
- 5 Functions (Reusing Code)
- 6 Function Arguments
- 7 Returning from Functions
- 8 Docstrings
- 9 Functions as Objects
- 10 Opening Files
- 11 Reading Files
- 12 Writing Files
- 13 Important Random Library Functions

Data structures[edit]

- Data Structure is a specialized format for organizing and storing efficiently.
- List is a linear data structure.
- linear data structure is a data structure in which you can add and delete items as and when there is a need(**Mutable**).

Refer to the python Docs for better over view of Data structures.

General Features of List[edit]

- for creating an empty list

```
List_name=[]
```

- Syntax for creating a list containing items

List_name=["item_1","item_2","item_3".....]

- Syntax for printing the contents of a list

print(List_name)

- Syntax for printing the contents of the list using for loop

for item in List_name: print(item) //This method is mainly used when we have to print the items one below the other in human friendly format. Note that item is an intuitive name.

- to add new data(*1 item only*) at the end of the list:

List_name.append(Data)

- to add new data(*list of items*) at the end of list :

List_name.extend(Data)

- to insert data at a desired position

List_name.insert(position,"item")//position starts from 0

- count function is used to count the number of occurrences of a particular item in a list
- Syntax

List_name.count(item)

- To find the number of items present in a list we use len function
- Syntax to find the length

len(List_name)

- to reverse the list:

List_name.reverse() #method1

List_name[: : -1] #method2

- to sort a list

List_name.sort() //sorts in ascending order List_name.sort(reverse=True) //sorts in descending order

- to count number of data of a particular type present in the list use:

shopping.count(data)//data=the object or item you want to search

- slicing of a list

shopping[1:4] /// displays shopping[1],[2],[3]

- to get the length of a list

len(shopping) // gives the length as output

- to get the maximum/minimmum value

max(shopping/ages)//returns the maximum value min(shopping/ages)//returns the minimum value

Week-3 Game programs[edit]

fizz buzz simple with if conditions[edit]

Problem: If a number is divisible by 3 the output should be fizz, if a number is divisible by 5 output should be buzz and if the number is divisible by both 3 and 5 then the output should be fizzbuzz.

```
n=int(input("enter your last point of your game!"))

for i in range (1,n):
    if(i%3==0 and i%5==0):
        print(i,"= fizzbuzz")
    else:
        if(i%3==0):
            print(i,"= fizz")
        else:
            if(i%5==0):
                print(i,"= buzz")
            else:
                print(i)
```

Crowd Computing Code[edit]

Posted By: Arunkumar | Doubts on this topic?

Below code is the default code from the Video.

Code:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from statistics import mean
from scipy import stats

Estimates = [1000,1010,1786,2000,1500,1500,100,120,150,
             150,150,170,175,180,197,200,200,200,200,200,
             200,220,220,220,220,234,250,250,250,250,250,
             250,250,250,250,263,270,275,275,286,300,300,
             300,300,300,300,300,300,320,350,350,350,400,
             400,400,400,400,450,467,500,500,500,500,500,
             500,500,550,550,600,600,600,650,700,700,720]

Estimates.sort()
m = stats.trim_mean(Estimates,0.1)
print("The near approximate value would be: ", m)
#Above two line are substitute for below lines, and "stats" where imported for above line.
#trm_val = int(0.1*(len(Estimates)))
#Estimates = Estimates[trm_val:]
#Estimates = Estimates[:len(Estimates)-trm_val]
#print(mean(Estimates))
```

To plot data using Python[edit]

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4],[1,4,9,16]) #To plot data on x and y axis respectively
plt.ylabel("some y values")  #To label y axis
plt.xlabel("some x values")  #To label x axis
```

Jumbled Words Game[edit]

Posted By: Arunkumar | Doubts on this topic? Ask by clicking here

This code is a customized and improvised code. I have added few extra functions like **clear screen**, **sleep**, and **3 chances**. I used functions everywhere possible, to reduce the length of the program. **And also, I have added a condition to random, so that we will never get a unjumbled output.**

Ask Any question related to this program in the Week-3 FAQs section, click here

```
import random
# import only system to know OS type and store in the variable "name"
from os import system, name
# import sleep to show output for some time period
from time import sleep

# This function is created to clear the screen
def wipeScreen():
    # Clear command for windows
    if name == 'nt':
        _ = system('cls')
    # Clear command for mac and linux
    else:
        _ = system('clear')
    # To add some empty vertical space from the top
    print('\n'*5)

def choose():
    cwords=['rainbow', 'dart', 'general', 'chemical', 'humble', 'family',
            'dramatic', 'gland', 'arm', 'arun', 'identical', 'mushroom', 'whale',
            'charm', 'comet', 'cultural', 'first', 'align', 'rich', 'India',
            'hump', 'fluid', 'bite', 'fighter', 'ten', 'photograph', 'escalator',
            'rose', 'dog', 'horse', 'jumble', 'python3.7', 'anaconda', 'spyder3.3.3',
```

```
24            'science', 'computer', 'programming', 'player',
'cheater']
25     pick=random.choice(cwords)
26     return pick
27
28 # This function is responsible for jumbling
29 def jumble(rec_word):
30     jumbled=rec_word
31     while (jumbled == rec_word):
32         jumbled="".join(random.sample(rec_word,len(rec_word)))
33     return jumbled
34
35 # This function is responsible for the final score.
36 def thank(p1n, p2n, s1, s2):
37     wipeScreen()
38     print(p1n, "Your final socre is: ", s1)
39     print(p2n, "Your final socre is: ", s2)
40     print("I hope you enjoyed playing. \nHave a nice day!")
41
42 # This function is responsible to check the user answer and
chances.
43 def j_game(player, score, pword, jword):
44     wipeScreen()
45     sleep(0.5)
46     print("\t"*3, player.capitalize(), "it's your turn!\n")
47     #Below print is just for decoration
48     print("*"*25)
49     print("Guess this word: ", jword)
50     for chance in range(1,4):
51         print("Chance", chance)
52         answer=str(input("- What is your guess: "))
53         if answer==pword:
54             score=score+1
55             print("\nYour scored", score, "point(s)!")
56             break
57         else:
```

```
58              print("Try again.")
59      print("\nCorrect answer is: ", pword)
60      #sleep for 2 seconds
61      sleep(2)
62      return score
63
64  def play():
65      wipeScreen()
66      sleep(0.5)
67      name_p1=str(input("Player 1, Please enter your name: "))
68      name_p2=str(input("Player 2, Please enter your name: "))
69      rounds=int(input("How many rounds you want to play? "))
70
71      # Cheating part. (Not yet updated/Added). I dont know English
much.
72      # So, i want this cheat.
73
74      # As this is a two player game, we need 2 while loop runs to
finish one round.
75      # So we are doubling the round count here.
76      rounds = rounds*2
77      turn=0
78      score_p1=0
79      score_p2=0
80
81      # Main game loop starts here
82      while (turn < rounds):
83          #Program chooses a random word from the above list
84          picked_word=choose()
85          #Picked word is jumbled here
86          jumbled_word=jumble(picked_word)
87
88          #Player1
89          if turn%2 == 0:
90              #We have created a j_game function. it will manage the
game part.
```

```
 91              score_p1=j_game(name_p1, score_p1, picked_word,
jumbled_word)
 92
 93          #Player2
 94          else:
 95              #We have created a j_game function. it will manage the
game part.
 96              score_p2=j_game(name_p2, score_p2, picked_word,
jumbled_word)
 97
 98          #turn increment
 99          turn=turn+1
100          if turn < rounds:
101              #Player choice to Quit
102              cont=int(input("Do you want to continue? \n press 1 to
continue or 0 to quit: "))
103              if cont == 0:
104                  break
105
106      #Exit greetings
107      thank(name_p1, name_p2, score_p1, score_p2)
108      # Main game loop ends here
109
110 # Entire game starts because of this below "play()" function call.
111 play()
```

Ask Any question related to this program in the Week-3 FAQs section, click here

Useful Functions & Methods for Lists[edit]

max(list) -- Returns the list item with the maximum value.

min(list) -- Returns the list item with minimum value.

list.count(obj) -- Returns a count of how many times an item occurs in a list.

list.remove(obj) -- Removes an object from a list.

list.reverse() -- Reverses objects in a list.

Functions (Reusing Code)[edit]

Code reuse is a very important part of programming in any language. Increasing code size at times makes it harder to maintain.

For a large programming project, it is essential to abide by the "**Don't Repeat Yourself**" or "**DRY**" principle. It makes the code easier to maintain.

Any statement that consists of a word followed by information in parenthesis is a **FUNCTION CALL**.

In addition to using pre-defined functions, we can create our own functions by using the "**def**" statement. The statements in the function are only executed when the function is called.

>>>def my_func():

```
        print("Hello")
```

The code block within every function starts with a colon and is indented. We MUST define functions before they are called.

Function Arguments[edit]

All the function definitions **MAY/ MAY NOT** take arguments.

Function arguments can be used as variables inside the function definition. However, they **cannot** be referenced outside of the function's definition. This also applies to other variables created inside the function.

Technically, "parameters" are the variables in a function definition and "arguments" are the values put into parameters when functions are called.

Returning from Functions[edit]

We use "**return**" statement.

The return statement cannot be used outside of a function definition.

Once you return a value from a function, it **immediately stops** being executed. Any code after the return statement will never happen.

Docstrings[edit]

Docstrings (documentation strings) serve a similar purpose to comments, as they are designed to explain code. However, they are more specific & have a different syntax.

Docstrings are created by putting a multiline string containing an explanation of the function below the function's first line.

>>>def shout(word):

```
        """
        Printing a word.
         ..........
        """
    print(word + '!')
```

>>> shout("Hello")

will give the output -- Hello !

Unlike conventional comments, docstrings are retained throughout the runtime of a program. This allows the programmer to inspect these comments at runtime.

Functions as Objects[edit]

Although they are created differently from normal variables, functions are just like any other kind of value.

They can be assigned & re-assigned to variables & later referenced by those names.

Functions can also be used as arguments of other functions.

Opening Files[edit]

Python can be used to read and write the contents of a file. Text files are easiest to manipulate. Before a file can be edited, it must be opened using the "**open**" function.

>>>myfile = open("filename.txt")

The argument of the open function is the path to the file. If the file is in the current working directory of the program, you can specify only its name.

"r" => read mode {Default}

"w" => write mode -- rewriting the contents of file.

"a" => append mode -- for adding new content to the end of file.

Adding "b" to a mode opens it in binary mode, which is used for non-text files(such as image and sound files).

"+" sign with each of the modes above give extra access to files.

Reading Files[edit]

The contents of a file that has been opened in text mode can be read using the "read()" method.

>>>file = open("filename.txt", "r")

>>>cont = file.read()

>>> print(cont)

>>>**file.close()**

The above code will print all the contents of the file.

To close the file: "close" method is used.

To read only a certain amount of a file, we can provide a number as an argument to read the function. This determines the number of bytes that should be read.

We can make more calls to read on the same file object to read more of the file byte by byte. With NO argument, read returns the rest of the file.

Just like passing no arguments, negative values will return the entire content.

After all the contents of a file have been read, any attempts to read further from that file will return an EMPTY STRING, because we are trying to read from the "end of file".

To retrieve each line in a file, we can use "readlines" method to return a list in which each element is a line in the file.

>>>**print(file.readlines())**

You can also use a for loop to iterate through the lines in the file.

>>>file = open("filename.txt", "r")

```
for line in file:
   print(line)
file.close()
```

In the output, the lines are separated by blank lines, as the print function automatically adds a new line at the end of its output.

Writing Files[edit]

To write to files, we use the "write" method, which writes a string to the file.

>>>file.write("Hello")

The "w" mode will create a file, if it does not already exist. When a file is opened in write mode, the file's existing content is deleted(content gets overridden).

The write() method returns the number of bytes written to a file, if successful.

To write something other than a string, it needs to be converted to a string first.

Important Random Library Functions[edit]

1) random.randint(1,5) -- Generates a random integer from 1 to 5.

2) random.randrange(1,5) -- Generates a random integer from 1 to 4.

3) random.random() -- Generates a random decimal number between 0 and 1.

4) random.choice(List_name) -- Generates an item randomly from the given list.

Lecture Notes:Week 4

From JOCWiki

Contents

[hide]

- 1 **Week-04 Programs and codes**
  - 1.1 **Magic Square**
    - 1.1.1 CODE
    - 1.1.2 Another way to generate a magic square of odd order
    - 1.1.3 Easier way to get the magic square for odd order by avoiding Rule 1 and 2
      - 1.1.3.1 Complete program is
    - 1.1.4 A brute force approach
  - 1.2 **PROGRAM FOR DOBBLE GAME:**
    - 1.2.1 METHOD 1:
    - 1.2.2 METHOD 2:
  - 1.3 **Birthday paradox**
    - 1.3.1 Chances of two person among 'n' persons have same birthday(using *import* math)
    - 1.3.2 Complete Program for Birthday paradox
- 2 **Notes**
  - 2.1 Modules
    - 2.1.1 The Standard Library & pip
    - 2.1.2 Major 3rd-Party Libraries
  - 2.2 Boolean Logic
  - 2.3 Break
  - 2.4 Continue
  - 2.5 List Slices
  - 2.6 List Comprehensions
  - 2.7 Map
  - 2.8 Useful String Functions
  - 2.9 Recursion
  - 2.10 Uses of sep/end in print function

Week-04 Programs and codes[edit]

Magic Square[edit]

A magic square is a square which is divided into smaller squares which enclose numbers. In that larger square **the sum of numbers in each row, column and diagonal is the same. The number starts from 1 and can go upto 's' where s is the number of small squares.**

Magic Square Hit and Trial 1

In this section , we shall deal with a square divided into smaller squares, each containing a number, such that the figures in each vertical, horizontal, and diagonal row add up to the same value .

Normally a magic square of 2*2 is not possible, which is the only exception.

What about a square containing only 1s at each cell?

So to start with, if we assume a 3x3 square then the sum of the elements must be equal to 15 in any direction - either in the vertical direction, horizontal direction or even diagonally . Generally for a n*n square , the value to which the sum should be equal to is called Magic Constant and is given by M = $n.(n^2+1)/2$.

Here for n = 3 , M =3.(9+1)/2 which is equal to 15 .

Some ways of arranging the elements in a 3*3 square to get a sum of 15 :

```
8  1  6        6  1  8        2  7  6
3  5  7        7  5  3        9  5  1
4  9  2        2  9  4        4  3  8
```

To solve these squares , there are few facts one must know :

- In any magic square 1 is located at the position: (n/2,n-1).
- If position of 1 is now taken as(p,q) then the position of 2 is at (p-1,q+1).
    1. If the calculated row position becomes equal to -1 then make it n-1 and if column position becomes n then make it 0.
    2. If the calculated position already contains a number, then decrease the column by 2 and increase the row by 1 .
    3. If anytime row position becomes-1 and column becomes n, switch to (0,n-2).

So first lets solve a 3 x 3 square using the facts mentioned.

Assume a 3 x 3 square as shown :

```
 E        E        E
(0,0)    (0,1)    (0,2)

 E        E        E
(1,0)    (1,1)    (1,2)

 E        E        E
(2,0)    (2,1)    (2,2)
```

Where "E" denotes the elements of the square and (a,b) represents the element's position.

Now a 3 x 3 magic square is shown :

```
 2          7          6
(0,0)      (0,1)      (0,2)


 9          5          1
(1,0)      (1,1)      (1,2)


 4          3          8
(2,0)      (2,1)      (2,2)
```

Steps:

1. Position of number 1 = (3/2, 3-1) = (1, 2)
2. Position of number 2 = (1-1,2+1) = (0,3) = (0, 0)
   Since the column value becomes n(i.e. 3) here,
   According to condition 1, make this column value equal to 0.
3. Position of number 3 = (0-1, 0+1) = (-1,1) = (3-1, 1) = (2, 1)
   Since the row value becomes -1 here,
   According to condition 1, make it equal to n-1 (i.e. 3-1)
4. Position of number 4 = (2-1, 1+1) = (1, 2)
   Since, at this position, 1 is there. So, apply condition 2.
   Incrementing the row by 1 and decrementing the column by 2, we get
   New position = (1+1,2-2)=(2,0)
5. Position of number 5=(2-1,0+1)=(1,1)
6. Position of number 6=(1-1,1+1)=(0,2)
7. Position of number 7 = (0-1, 2+1) = (-1,3)
   Since, the row value becomes -1 and column value becomes n(i.e 3) here,
   According to condition 3, switching row value to 0 and column value to n-2(i.e. 3-2), we get
   New position = (0, 3-2) = (0,1)
8. Position of number 8=(0-1,1+1)=(-1,2)=(2,2)
   Since, row value becomes -1 here,
   According to condition 1, make it equal to n-1(i.e. 3-1)
9. Position of number 9=(2-1,2+1)=(1,3)=(1,0)
   Since, column value becomes n (i.e 3) here,
   According to condition 1, make it equal to 0.

We see that this method consumes a lot of time. As someone once said:"First solve the problem, then write the code". So, we shall see if a piece of code employing the facts can make us solve this puzzle faster.

We shall write two codes - One for generating an odd sized magic square and the other for printing the magic square :

CODE[edit]

```
1 #generating magic square
2 def generateSquare(n):
3
```

```
    # 2-D array with all slots set to 0
    magicSquare = [[0 for x in range(n)]for y in range(n)]

    # initialize position of 1
    i = n//2
    j = n-1

    # Fill the magic square by placing values
    num = 1
    while num <= (n*n):
        if i == -1 and j == n: # third condition
            j = n-2
            i = 0 #can use multiple assignment i,j = 0,n-2
        else:
            # next number goes out of right side of square
            if j == n:
                j = 0
            # next number goes out of upper side
            if i < 0:
                i = n-1

        if magicSquare[i][j]: # 2nd condition
            j = j-2
            i = i+1
            continue
        else:
            magicSquare[i][j] = num
            num = num+1

        j = j+1
        i = i-1 #1st

# Printing magic square

    print ("Magic Squre for n = ",n)
    print ("Sum of each row or column",n*(n*n+1)/2)
```

```
40
41     for i in range(0,n):
42         for j in range(0,n):
43
44             print (magicSquare[i][j],end=" ")
45         print()
46 # Works only when n is odd
47 n = 7
48 generateSquare(n)
```

```
Output: One should change the value of n defined in the program
to get magic square of
 that particular order.

The Magic Square for n=7:
Sum of each row or column 175:

 20  12   4  45  37  29  28
 11   3  44  36  35  27  19
  2  43  42  34  26  18  10
 49  41  33  25  17   9   1
 40  32  24  16   8   7  48
 31  23  15  14   6  47  39
 22  21  13   5  46  38  30
```

#Top

Another way to generate a magic square of odd order[edit]

```
 1 import numpy as np
 2
 3 def magic_square(n):
 4     arr=np.zeros((n,n))
 5     p=n//2
 6     q=n-1
 7     a=1
 8     arr[p][q]=a
 9     for i in range(n**2-1):
10         p=p-1
11         q=q+1
```

```
        a=a+1
        if (p!=-1) and (q!=n):
            if (arr[p][q]==0):
                arr[p][q]=a
            else:
                p=p+1
                q=q-2
                arr[p][q]=a
        elif (q==n):
            q=0
            if (arr[p][q]==0):
                arr[p][q]=a
            else:
                p=p+1
                q=q-2
                arr[p][q]=a
        elif(p==-1):
            p=n-1
            if (arr[p][q]==0):
                arr[p][q]=a
            else:
                p=p+1
                q=q-2
                arr[p][q]=a
        elif(p==-1) and (q==n):
            p=n-1
            q=0
            if (arr[p][q]==0):
                arr[p][q]=a
            else:
                p=p+1
                q=q-2
                arr[p][q]=a
    return arr
n=input("enter the value of n \a");n=int(n)
a=magic_square(n)
```

```
48 print(a)
```
#Top

Easier way to get the magic square for odd order by avoiding Rule 1 and 2[edit]

This is a modification to above program(Magic square for odd order) to **AVOID Rule 1 and 2**

First notice the important property of magic squares: "A magic square remains a magic square if it are rotated". This is intuitively obvious.

So if we rotate left 90 deg (n//2,n-1) becomes (0,n//2) which is 1's position

Now the rule to construct magic square is from current position just go UP then RIGHT,

```
i=i-1

j=j+1
```

To avoid -1 or n which is out of index error in list, we use this TRICK (Rule 1 and 2 avoided):

```
i=(i-1+n)%n

j=(j+1+n)%n
```

this always puts i,j in proper range for list index

actually +n can be avoided for calculating j since j+1 is always positive

Then put the next number there if it does not contain any other value (==0) ,

if this new position is already occupied put the next value below the previous position (i=i_previous+1) Now there are 2 approaches

```
         1) use a temp. variable to store previous values of i
and j
         2) Use the same trick again to revert with a small
change
                      i=(i+1+n)%n                  # +n can be
avoided since i+1 is always positive

                      j=(j-1+n)%n
```

now increase i

```
i=i+1  # going DOWN
then put next value
```

Complete program is[edit]

```
1 def printarr(A):
2     print("\n")
3     for i in range(0,n):
4        for j in range(0,n):
```

```
            print (A[i][j],end=" ")
        print()
    print("\n")


n=int(input("Enter order of magic square "))
magicSquare=[[0 for x in range(n)]for y in range(n)]
i=0
j=n//2
num=1

while num<=n*n:
    if magicSquare[i][j]==0:        #if unfilled
        magicSquare[i][j]=num

    else:                           #if filled
        i=(i+1+n)%n                 #revert
        j=(j-1+n)%n
        i=(i+1)%n                       #go down
        magicSquare[i][j]=num       #update next value

    i=(i-1+n)%n                     #go UP
    j=(j+1+n)%n                     #go DOWN
    num+=1

printarr(magicSquare)
print("Sum of each row, column and diagonals = ",n*(n*n+1)//2)
```

Output:

```
Enter order of magic square 7


30 39 48  1 10 19 28
38 47  7  9 18 27 29
46  6  8 17 26 35 37
 5 14 16 25 34 36 45
13 15 24 33 42 44  4
21 23 32 41 43  3 12
22 31 40 49  2 11 20
```

```
Sum of each row, column and diagonals =  175
```

#Top

A brute force approach[edit]

CODE:

```
import random
a=[1,2,3,4,5,6,7,8,9]
while(1):
    random.shuffle(a)
    if((a[0]+a[1]+a[2])==15 and (a[3]+a[4]+a[5])==15 and
       (a[6]+a[7]+a[8])==15 and (a[0]+a[3]+a[6])==15 and
       (a[1]+a[4]+a[7])==15 and (a[2]+a[5]+a[8])==15 and
       (a[0]+a[4]+a[8])==15 and (a[2]+a[4]+a[6])==15):
        break
for i in range(9):
    print(a[i],end=" ")
    if(i%3==2):
        print("")
```

#Top

PROGRAM FOR DOBBLE GAME:[edit]

METHOD 1:[edit]

```
import string
import random
symbol = list(string.ascii_letters)
card1 = [0]*5
card2 = [0]*5
pos1 = random.randrange(0,5)
pos2 = random.randint(0,4)
samesymbol  = random.choice(symbol)
symbol.remove(samesymbol)
card1[pos1]=samesymbol
card2[pos2]=samesymbol
if pos1 == pos2 :
```

```
    card1[pos1] = samesymbol
    card2[pos1] = samesymbol
else:
    card1[pos1] = samesymbol
    card2[pos2] = samesymbol
    card1[pos2] = random.choice(symbol)
    symbol.remove(card1[pos2])
    card2[pos1] = random.choice(symbol)
    symbol.remove(card2[pos1])
for i in range(5):
    if i != pos1 and i != pos2 :
        card1[i]=random.choice(symbol)
        symbol.remove(card1[i])
        card2[i]=random.choice(symbol)
        symbol.remove(card2[i])

print(card1,card2)

ans = input('Guess the common symbol')
if ans == samesymbol:
    print("Hurrah!")
else:
    print("OOPS")
```

METHOD 2:[edit]

```
import string
import random
symbol = list(string.ascii_letters)
card1 = [0]*5
card2 = [0]*5
pos1 = random.randrange(0,5)
pos2 = random.randint(0,4)
ss  = random.choice(symbol)
```

```
 9 symbol.remove(ss)
10 card1[pos1]=ss
11 card2[pos2]=ss
12 for i in range(5):
13     if i != pos1:
14         card1[i]=random.choice(symbol)
15         symbol.remove(card1[i])
16 for i in range(5):
17     if i != pos2:
18         card2[i]=random.choice(symbol)
19         symbol.remove(card2[i])
20
21 print(card1,card2)
22
23 ans = input('Guess the common symbol')
24 if ans == ss:
25     print("Hurrah!")
26 else:
27     print("OOPS")
```

#Top

Birthday paradox[edit]

Chances of two person among 'n' persons have same birthday(using *import* math)[edit]

Let the probability that two people in a room with 'n' number of people have same birthday be P(same).

Let the probability that two people in a room with 'n' number of people have different birthday be P(different)

P(same) = 1 – P(different)

P(different) can be written as 1 x (364/365) x (363/365) x (362/365) x …. x (1 – (n-1)/365) .....(1)

The n'th person should have a birthday which is not same as any of the earlier considered (n-1) persons.

To find the generalized formula, I'm using Taylor's series. (*there could be other ways also*)

The Approximation expression (1):-

```
e^x = 1 + x + (x^2/2!) + ....
```

first-order approximation for e^x for x<<1:

```
e^x ~ 1 + x
```
now, x = -a/365

```
e^(-a/365) ~ 1 - a/365
```
The above expression derived for p(different) can be written as 1 x (1 – 1/365) x (1 – 2/365) x (1 – 3/365) x …. x (1 – (n-1)/365)

By putting the value of 1 – a/365 as e^(-a/365), we get following.

```
~ 1 x e^(-1/365) x e^(-2/365)... x e^(-n+1/365)
~ 1 x e^((-(n(n-1))/2)/365)
```
Therefore,

p(same) = 1 - p(different)

```
 p(same) ~ 1 - e^((-n^2)/(2x365))
```

Code:-

```
import math

def find( n ):
    return ((1-math.exp(-n**2/(2*365)))*100)

n=int(input("Enter the total number of people\n"))
if n<2:
    print("No chance")
else:
    print(find(n),"% chance")
```
.

Complete Program for Birthday paradox[edit]

```
import random
birthlist = []
for i in range(30):
    year = random.randint(1993,2018)
    ly = 0
    if (year%100 !=0 and year%4 == 0 or year%400==0):
        ly = 1
    month = random.randint(1,12)
    if (ly == 1 and month == 2):
        date = random.randint(1,29)
    if (ly == 0 and month == 2):
        date = random.randint(1,28)
    elif (month % 2 == 0 and month<7):
        date = random.randint(1,30)
```

```
    elif (month % 2 == 0 and month>7):
        date = random.randint(1,31)
    elif (month % 2 != 0 and month<=7):
        date = random.randint(1,31)
    elif (month % 2 != 0 and month>7):
        date = random.randint(1,30)
    dd = (date,month) #Tuples
    birthlist.append(dd) #List of tuples

print(birthlist)
#Checking the number of matches i.e people having same birthday
match = []
for i in birthlist:
    j = 0
    while i in birthlist:
        birthlist.remove(i)
        j += 1
    match.append(j)
count = 0
for i in match:
    if i > 1:
        count += 1
print('Number of matches are',count)
```

#Top

Notes[edit]

Modules[edit]

Modules are pieces of code that others have written to fulfill common tasks, such as generating random numbers, performing mathematical operations etc.

The basic way to use a module is to add "**import module_name**" at the top of your code & then use "**module_name.var**" to access functions & values with the name "var" in the module.

```
>>> import random
```

```
>>> for i in range(5):

print(random.randint(1,6))
```

The above code uses "randint" function defined in the random module to print 5 random numbers in the range 1 to 6 (including both 1 and 6).

There is another kind of import that can be used if you only need certain functions from a module. These take the form "**from module_name import var**" & then var can be used as if it were defined normally on your code.

```
>>> from math import pi

>>> print(pi)
```

Use a comma separated list to import multiple objects.

```
>>> from math import pi, sqrt
```

The asterisk (*) imports all objects from a module.

```
>>> from math import *
```

Trying to import a module that isn't available causes an "**ImportError**"

You can import a module or object under a different name using the "as" keyword. This is mainly used when a module or object has a long/confusing name.

```
>>> from math import sqrt as square_root
```

The Standard Library & pip[edit]

There are three main types of modules in python:

1. Those you write yourself
2. Those you install from external sources
3. Those that are pre-installed with python

The last type is called the standard library and contains many useful modules.

Some of the standard library's useful modules include -- string, re, datetime, math, random, os, multiprocessing, subprocess, socket, email, json, doctest and many more.

Tasks that can be done by the standard library include string parsing, data serialization testing, debugging & manipulating dates, emails, command line arguments & much more !!

Some of the modules in the standard library are written in python and some are written in C. (please visit www.python.org for more information).

Major 3rd-Party Libraries[edit]

The python standard library alone contains extensive functionality. However, some tasks require the use of third-party libraries.

Django: The most frequently used web framework written in python, Django powers websites that include Instagram and Disqus. It has many useful features, and whatever features it lacks are covered by extension packages.

CherryPy & Flask are also popular web frameworks.

For scraping data from websites, the library **BeautifulSoup** is very useful, and leads to better results than building your own scrapper with regular expressions.

While Python does offer modules for programmatically accessing websites, such as "urllib", they are quite cumbersome to use. Third-party library requests make it much easier to use HTTP requests.

A number of third-party modules are available that make it much easier to carry out scientific and mathematical computing with Python.

The module **matplotlib** allows you to create graphs based on data in Python.

The module **NumPy** allows for the use of multidimensional arrays that are much faster than the native python solution of nested lists. It also contains functions to perform mathematical operations such as matrix transformations on the arrays.

The library **SciPy** contains numerous extensions to th functionality of NumPy.

Python can also be used for **game development**. Usually, it is used as a scripting language for games written in other languages, but it can be used to make games by itself.

Boolean Logic[edit]

Two Boolean values: True & False

Can be compared using the equal to operator == (Comparison Operator), not equal operator (!=), < or >, >= or <=

Greater than & smaller than operators can also be used to compare strings lexicographically (The alphabetical order of words is based on the alphabetical order of their component letters).

Boolean logic is used to make more complicated conditions for if statements that rely on more than one condition.

Python's Boolean operators are "**and**", "**or**" & "**not**". Python **uses words** for its Boolean operators.

Break[edit]

To end a loop pre-maturely, the break statement can be used. When encountered in a loop, the break statement causes the loop to finish immediately.

Using the break statement OUTSIDE THE LOOP CAUSES AN ERROR.

Continue[edit]

Unlike break, continue jumps back to the top of the loop, rather than stopping it.

Basically, the continue statement stops the current iteration & continues with the next one.

Using the continue statement OUTSIDE THE LOOP CAUSES AN ERROR.

List Slices[edit]

List slices provide a more advanced way of retrieving values from a list. Basic list slicing involves **indexing a list with two colon separated integers**. This **returns a new list** containing all the values in the old list between the specified indices.

>>>squares = [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

>>>print(squares[2:6])

Output: [4, 9, 16, 25]

Like the arguments to range, the first index provided in a slice is included in the result but the second isn't. If the first number in a slice is omitted, it is taken to be the start of list. If the second number is omitted, it is taken to be the end of the list.

Slicing can also be done on tuples.

List slices can also have a third argument, representing the step, to include only alternate value sin the slice.

Negative values can be used in list slicing as well as normal indexing. When negative values are used for the first & second values in a slice, they count from the end of the list.

If the negative value is used for the step, the slice is done backwards. using [::-1] is common & idiomatic way to REVERSE a list.

List Comprehensions[edit]

List comprehensions are a useful way of quickly creating lists whose contents obey a simple rule.

>>>cubes = [i3 for i in range(5)]**

>>>print(cubes)

List comprehensions are **inspired by set-builder notation** in mathematics.

A list comprehension can also contain an if statement to enforce a condition on values in the list.

>>>evens = [i2 for i in range(10) if i**2 % 2 == 0]**

>>>print(evens)

Output: [0, 4, 16, 36, 64]

Trying to create a list in a very extensive range will result in a MemoryError. This issue is solved by "generators".

Map[edit]

The built-in function map is a very useful higher order function that operates on lists (or similar objects called iterables)

The function map takes a function & an iterable as arguments, and returns a new iterable with a function applied to each argument.

```
>>>def add(x):
    return(x+5)
>>>nums = [11, 22, 33, 44, 55]
>>>result = list(map(add, nums))
>>>print(result)
```

Output: [16, 27, 38, 49, 60]

Useful String Functions[edit]

join -- Joins a list of strings with another string as a separation.

replace -- Replaces one substring in a string with another.

startswith & **endswith** -- Determines if there is a substring at the start and end of a string respectively.

To change the case of a string we use **lower()** & **upper()**

The method "split" is the opposite of join, turning a string with a certain separator into a list.

```
>>> print(".".join(["hello","world"]))
--> Output: hello.world

>>> print("hello me".replace("me", "world"))
--> Output: hello world

>>> print("This world".startswith("This"))
--> Output: True

>>> print("hello".upper())
--> Output: HELLO
```

```
>>> print("spam", "hello", "world")

--> Output: spam hello world
```

#Top

Recursion[edit]

The fundamental part of recursion is self reference -- functions calling itself.

The base case acts as the exit condition of the recursion.

Recursion can also be indirect. One function can call a second, which calls the first, which calls the second and so on... This can occur with any number of functions.

```
>>>def is_even(x):

        if x == 0:
              return(True)
        else:
              return(is_odd(x-1))
>>>def is_odd(x):

        return(not is_even(x))
```

>>>print(is_odd(17)) --> True

>>>print(is_even(23)) --> False

To find the factorial of a number, recursion technique is used extensively

```
>>>def factorial(n):

    if n == 1:
        return 1
    return n*factorial(n-1)
```

>>>print (factorial(5)) --> 120

Uses of sep/end in print function[edit]

The print() function prints the specified message to the screen, or other standard output device. The message can be a string, or any other object, the object will be converted into a string before written to the screen.

The function "sep" stands for separation. It is used to 'separate' multiple values of print statement with the provided argument

1.Example:

```
>>> print("Hello", "how are you?", sep = ",")

>>> Output : Hello, how are you?
```

2.Example: >>> x = "apple"

>>> y = "banana"

>>> z = "cherry"

>>> print(x,y,z, sep = ", ")

>>> Output : apple, banana, cherry

The function "end" specifies what to print at the end. By default a new line is inserted at the end.

1.Example:

>>> print("Python is fun", end = ".")

>>> Output : Python is fun.

2.Example:

>>> print("Hello", "how are you", sep = ", ",end = "?")

>>>Output: Hello, how are you?

Lecture Notes:Week 4

From JOCWiki

Contents

[hide]

- 1 **Week-04 Programs and codes**
  - 1.1 **Magic Square**
    - 1.1.1 CODE
    - 1.1.2 Another way to generate a magic square of odd order
    - 1.1.3 Easier way to get the magic square for odd order by avoiding Rule 1 and 2
      - 1.1.3.1 Complete program is
    - 1.1.4 A brute force approach
  - 1.2 **PROGRAM FOR DOBBLE GAME:**
    - 1.2.1 METHOD 1:
    - 1.2.2 METHOD 2:
  - 1.3 **Birthday paradox**
    - 1.3.1 Chances of two person among 'n' persons have same birthday(using *import* math)
    - 1.3.2 Complete Program for Birthday paradox
- 2 **Notes**
  - 2.1 Modules
    - 2.1.1 The Standard Library & pip
    - 2.1.2 Major 3rd-Party Libraries
  - 2.2 Boolean Logic
  - 2.3 Break
  - 2.4 Continue
  - 2.5 List Slices
  - 2.6 List Comprehensions
  - 2.7 Map
  - 2.8 Useful String Functions

- 2.9 Recursion
- 2.10 Uses of sep/end in print function

Week-04 Programs and codes[edit]

Magic Square[edit]

A magic square is a square which is divided into smaller squares which enclose numbers. In that larger square **the sum of numbers in each row, column and diagonal is the same. The number starts from 1 and can go upto 's' where s is the number of small squares.**

Magic Square Hit and Trial 1

In this section , we shall deal with a square divided into smaller squares, each containing a number, such that the figures in each vertical, horizontal, and diagonal row add up to the same value .

Normally a magic square of 2*2 is not possible, which is the only exception.

What about a square containing only 1s at each cell?

So to start with, if we assume a 3x3 square then the sum of the elements must be equal to 15 in any direction - either in the vertical direction, horizontal direction or even diagonally . Generally for a n*n square , the value to which the sum should be equal to is called Magic Constant and is given by M = $n.(n^2+1)/2$.

Here for n = 3 , M =3.(9+1)/2 which is equal to 15 .

Some ways of arranging the elements in a 3*3 square to get a sum of 15 :

```
8  1  6        6  1  8        2  7  6
3  5  7        7  5  3        9  5  1
4  9  2        2  9  4        4  3  8
```

To solve these squares , there are few facts one must know :

- In any magic square 1 is located at the position: (n/2,n-1).
- If position of 1 is now taken as(p,q) then the position of 2 is at (p-1,q+1).
    1. If the calculated row position becomes equal to -1 then make it n-1 and if column position becomes n then make it 0.
    2. If the calculated position already contains a number, then decrease the column by 2 and increase the row by 1 .
    3. If anytime row position becomes-1 and column becomes n, switch to (0,n-2).

So first lets solve a 3 x 3 square using the facts mentioned.

Assume a 3 x 3 square as shown :

```
    E          E          E
```

```
(0,0)     (0,1)     (0,2)


 E         E         E
(1,0)     (1,1)     (1,2)


 E         E         E
(2,0)     (2,1)     (2,2)
```

Where "E" denotes the elements of the square and (a,b) represents the element's position.

Now a 3 x 3 magic square is shown :

```
 2         7         6
(0,0)     (0,1)     (0,2)


 9         5         1
(1,0)     (1,1)     (1,2)


 4         3         8
(2,0)     (2,1)     (2,2)
```

Steps:

1. Position of number 1 = (3/2, 3-1) = (1, 2)
2. Position of number 2 = (1-1,2+1) = (0,3) = (0, 0)
   Since the column value becomes n(i.e. 3) here,
   According to condition 1, make this column value equal to 0.
3. Position of number 3 = (0-1, 0+1) = (-1,1) = (3-1, 1) = (2, 1)
   Since the row value becomes -1 here,
   According to condition 1, make it equal to n-1 (i.e. 3-1)
4. Position of number 4 = (2-1, 1+1) = (1, 2)
   Since, at this position, 1 is there. So, apply condition 2.
   Incrementing the row by 1 and decrementing the column by 2, we get
   New position = (1+1,2-2)=(2,0)
5. Position of number 5=(2-1,0+1)=(1,1)
6. Position of number 6=(1-1,1+1)=(0,2)
7. Position of number 7 = (0-1, 2+1) = (-1,3)
   Since, the row value becomes -1 and column value becomes n(i.e 3) here,
   According to condition 3, switching row value to 0 and column value to n-2(i.e. 3-2), we get
   New position = (0, 3-2) = (0,1)
8. Position of number 8=(0-1,1+1)=(-1,2)=(2,2)
   Since, row value becomes -1 here,
   According to condition 1, make it equal to n-1(i.e. 3-1)
9. Position of number 9=(2-1,2+1)=(1,3)=(1,0)

```
  Since, column value becomes n (i.e 3) here,
  According to condition 1, make it equal to 0.
```

We see that this method consumes a lot of time. As someone once said:"First solve the problem, then write the code". So, we shall see if a piece of code employing the facts can make us solve this puzzle faster.

We shall write two codes - One for generating an odd sized magic square and the other for printing the magic square :

CODE[edit]

```
#generating magic square
def generateSquare(n):

    # 2-D array with all slots set to 0
    magicSquare = [[0 for x in range(n)]for y in range(n)]

    # initialize position of 1
    i = n//2
    j = n-1

    # Fill the magic square by placing values
    num = 1
    while num <= (n*n):
        if i == -1 and j == n: # third condition
            j = n-2
            i = 0 #can use multiple assignment i,j = 0,n-2
        else:
            # next number goes out of right side of square
            if j == n:
                j = 0
            # next number goes out of upper side
            if i < 0:
                i = n-1

        if magicSquare[i][j]: # 2nd condition
            j = j-2
            i = i+1
            continue
```

```
29          else:
30              magicSquare[i][j] = num
31              num = num+1
32
33          j = j+1
34          i = i-1 #1st
35
36 # Printing magic square
37
38      print ("Magic Squre for n = ",n)
39      print ("Sum of each row or column",n*(n*n+1)/2)
40
41      for i in range(0,n):
42          for j in range(0,n):
43
44              print (magicSquare[i][j],end=" ")
45          print()
46 # Works only when n is odd
47 n = 7
48 generateSquare(n)
```

```
Output: One should change the value of n defined in the program
to get magic square of
 that particular order.

The Magic Square for n=7:
Sum of each row or column 175:

 20  12   4  45  37  29  28
 11   3  44  36  35  27  19
  2  43  42  34  26  18  10
 49  41  33  25  17   9   1
 40  32  24  16   8   7  48
 31  23  15  14   6  47  39
 22  21  13   5  46  38  30
```

#Top

Another way to generate a magic square of odd order[edit]

```
import numpy as np

def magic_square(n):
    arr=np.zeros((n,n))
    p=n//2
    q=n-1
    a=1
    arr[p][q]=a
    for i in range(n**2-1):
        p=p-1
        q=q+1
        a=a+1
        if (p!=-1) and (q!=n):
            if (arr[p][q]==0):
                arr[p][q]=a
            else:
                p=p+1
                q=q-2
                arr[p][q]=a
        elif (q==n):
            q=0
            if (arr[p][q]==0):
                arr[p][q]=a
            else:
                p=p+1
                q=q-2
                arr[p][q]=a
        elif(p==-1):
            p=n-1
            if (arr[p][q]==0):
                arr[p][q]=a
            else:
                p=p+1
                q=q-2
                arr[p][q]=a
```

```
36          elif(p==-1) and (q==n):
37              p=n-1
38              q=0
39              if (arr[p][q]==0):
40                  arr[p][q]=a
41              else:
42                  p=p+1
43                  q=q-2
44                  arr[p][q]=a
45      return arr
46 n=input("enter the value of n \a");n=int(n)
47 a=magic_square(n)
48 print(a)
```

#Top

Easier way to get the magic square for odd order by avoiding Rule 1 and 2[edit]

This is a modification to above program(Magic square for odd order) to **AVOID Rule 1 and 2**

First notice the important property of magic squares: "A magic square remains a magic square if it are rotated". This is intuitively obvious.

So if we rotate left 90 deg (n//2,n-1) becomes (0,n//2) which is 1's position

Now the rule to construct magic square is from current position just go UP then RIGHT,

```
i=i-1

j=j+1
```

To avoid -1 or n which is out of index error in list, we use this TRICK (Rule 1 and 2 avoided):

```
i=(i-1+n)%n

j=(j+1+n)%n
```

this always puts i,j in proper range for list index

actually +n can be avoided for calculating j since j+1 is always positive

Then put the next number there if it does not contain any other value (==0) ,

if this new position is already occupied put the next value below the previous position (i=i_previous+1) Now there are 2 approaches

```
        1) use a temp. variable to store previous values of i
and j
```

```
        2) Use the same trick again to revert with a small
change
                    i=(i+1+n)%n                    # +n can be
avoided since i+1 is always positive

                    j=(j-1+n)%n
```
now increase i

```
i=i+1  # going DOWN
then put next value
```

Complete program is[edit]

```
def printarr(A):
    print("\n")
    for i in range(0,n):
       for j in range(0,n):
               print (A[i][j],end=" ")
       print()
    print("\n")


n=int(input("Enter order of magic square "))
magicSquare=[[0 for x in range(n)]for y in range(n)]
i=0
j=n//2
num=1

while num<=n*n:
    if magicSquare[i][j]==0:        #if unfilled
        magicSquare[i][j]=num

    else:                           #if filled
        i=(i+1+n)%n                 #revert
        j=(j-1+n)%n
        i=(i+1)%n                       #go down
        magicSquare[i][j]=num           #update next value

    i=(i-1+n)%n                     #go UP
    j=(j+1+n)%n                     #go DOWN
```

```
28      num+=1
29
30 printarr(magicSquare)
31 print("Sum of each row, column and diagonals = ",n*(n*n+1)//2)
```

```
Output:
Enter order of magic square 7


30 39 48  1 10 19 28
38 47  7  9 18 27 29
46  6  8 17 26 35 37
 5 14 16 25 34 36 45
13 15 24 33 42 44  4
21 23 32 41 43  3 12
22 31 40 49  2 11 20

Sum of each row, column and diagonals =  175
```

#Top

A brute force approach[edit]

CODE:

```
 1 import random
 2 a=[1,2,3,4,5,6,7,8,9]
 3 while(1):
 4     random.shuffle(a)
 5     if((a[0]+a[1]+a[2])==15 and (a[3]+a[4]+a[5])==15 and
 6        (a[6]+a[7]+a[8])==15 and (a[0]+a[3]+a[6])==15 and
 7        (a[1]+a[4]+a[7])==15 and (a[2]+a[5]+a[8])==15 and
 8        (a[0]+a[4]+a[8])==15 and (a[2]+a[4]+a[6])==15):
 9         break
10 for i in range(9):
11     print(a[i],end=" ")
12     if(i%3==2):
13         print("")
```

#Top

PROGRAM FOR DOBBLE GAME:[edit]

METHOD 1:[edit]

```
import string
import random
symbol = list(string.ascii_letters)
card1 = [0]*5
card2 = [0]*5
pos1 = random.randrange(0,5)
pos2 = random.randint(0,4)
samesymbol  = random.choice(symbol)
symbol.remove(samesymbol)
card1[pos1]=samesymbol
card2[pos2]=samesymbol
if pos1 == pos2 :
    card1[pos1] = samesymbol
    card2[pos1] = samesymbol
else:
    card1[pos1] = samesymbol
    card2[pos2] = samesymbol
    card1[pos2] = random.choice(symbol)
    symbol.remove(card1[pos2])
    card2[pos1] = random.choice(symbol)
    symbol.remove(card2[pos1])
for i in range(5):
    if i != pos1 and i != pos2 :
        card1[i]=random.choice(symbol)
        symbol.remove(card1[i])
        card2[i]=random.choice(symbol)
        symbol.remove(card2[i])

print(card1,card2)

ans = input('Guess the common symbol')
if ans == samesymbol:
    print("Hurrah!")
else:
    print("OOPS")
```

METHOD 2:[edit]

```
import string
import random
symbol = list(string.ascii_letters)
card1 = [0]*5
card2 = [0]*5
pos1 = random.randrange(0,5)
pos2 = random.randint(0,4)
ss  = random.choice(symbol)
symbol.remove(ss)
card1[pos1]=ss
card2[pos2]=ss
for i in range(5):
    if i != pos1:
        card1[i]=random.choice(symbol)
        symbol.remove(card1[i])
for i in range(5):
    if i != pos2:
        card2[i]=random.choice(symbol)
        symbol.remove(card2[i])

print(card1,card2)

ans = input('Guess the common symbol')
if ans == ss:
    print("Hurrah!")
else:
    print("OOPS")
```

#Top

Birthday paradox[edit]

Chances of two person among 'n' persons have same birthday(using *import* math)[edit]

Let the probability that two people in a room with 'n' number of people have same birthday be P(same).

Let the probability that two people in a room with 'n' number of people have different birthday be P(different)

P(same) = 1 – P(different)

P(different) can be written as 1 x (364/365) x (363/365) x (362/365) x …. x (1 – (n-1)/365) .....(1)

The n'th person should have a birthday which is not same as any of the earlier considered (n-1) persons.

To find the generalized formula, I'm using Taylor's series. (*there could be other ways also*)

The Approximation expression (1):-

```
e^x = 1 + x + (x^2/2!) + ....
```

first-order approximation for e^x for x<<1:

```
e^x ~ 1 + x
```

now, x = -a/365

```
e^(-a/365) ~ 1 - a/365
```

The above expression derived for p(different) can be written as 1 x (1 – 1/365) x (1 – 2/365) x (1 – 3/365) x …. x (1 – (n-1)/365)

By putting the value of 1 – a/365 as e^(-a/365), we get following.

```
~ 1 x e^(-1/365) x e^(-2/365)... x e^(-n+1/365)
~ 1 x e^((-(n(n-1))/2)/365)
```

Therefore,

p(same) = 1 - p(different)

```
 p(same) ~ 1 - e^((-n^2)/(2x365))
```

Code:-

```
import math

def find( n ):
    return ((1-math.exp(-n**2/(2*365)))*100)

n=int(input("Enter the total number of people\n"))
if n<2:
    print("No chance")
else:
    print(find(n),"% chance")
```

.

Complete Program for Birthday paradox[edit]

```
 1 import random
```

```
birthlist = []
for i in range(30):
    year = random.randint(1993,2018)
    ly = 0
    if (year%100 !=0 and year%4 == 0 or year%400==0):
        ly = 1
    month = random.randint(1,12)
    if (ly == 1 and month == 2):
        date = random.randint(1,29)
    if (ly == 0 and month == 2):
        date = random.randint(1,28)
    elif (month % 2 == 0 and month<7):
        date = random.randint(1,30)
    elif (month % 2 == 0 and month>7):
        date = random.randint(1,31)
    elif (month % 2 != 0 and month<=7):
        date = random.randint(1,31)
    elif (month % 2 != 0 and month>7):
        date = random.randint(1,30)
    dd = (date,month) #Tuples
    birthlist.append(dd) #List of tuples

print(birthlist)
#Checking the number of matches i.e people having same birthday
match = []
for i in birthlist:
    j = 0
    while i in birthlist:
        birthlist.remove(i)
        j += 1
    match.append(j)
count = 0
for i in match:
    if i > 1:
        count += 1
```

```
37 print('Number of matches are',count)
```

#Top

Notes[edit]

Modules[edit]

Modules are pieces of code that others have written to fulfill common tasks, such as generating random numbers, performing mathematical operations etc.

The basic way to use a module is to add "**import module_name**" at the top of your code & then use "**module_name.var**" to access functions & values with the name "var" in the module.

```
>>> import random

>>> for i in range(5):

print(random.randint(1,6))
```

The above code uses "randint" function defined in the random module to print 5 random numbers in the range 1 to 6 (including both 1 and 6).

There is another kind of import that can be used if you only need certain functions from a module. These take the form "**from module_name import var**" & then var can be used as if it were defined normally on your code.

```
>>> from math import pi

>>> print(pi)
```

Use a comma separated list to import multiple objects.

```
>>> from math import pi, sqrt
```

The asterisk (*) imports all objects from a module.

```
>>> from math import *
```

Trying to import a module that isn't available causes an "**ImportError**"

You can import a module or object under a different name using the "as" keyword. This is mainly used when a module or object has a long/confusing name.

```
>>> from math import sqrt as square_root
```

The Standard Library & pip[edit]

There are three main types of modules in python:

1. Those you write yourself
2. Those you install from external sources

3. Those that are pre-installed with python

The last type is called the standard library and contains many useful modules.

Some of the standard library's useful modules include -- string, re, datetime, math, random, os, multiprocessing, subprocess, socket, email, json, doctest and many more.

Tasks that can be done by the standard library include string parsing, data serialization testing, debugging & manipulating dates, emails, command line arguments & much more !!

Some of the modules in the standard library are written in python and some are written in C. (please visit www.python.org for more information).

Major 3rd-Party Libraries[edit]

The python standard library alone contains extensive functionality. However, some tasks require the use of third-party libraries.

Django: The most frequently used web framework written in python, Django powers websites that include Instagram and Disqus. It has many useful features, and whatever features it lacks are covered by extension packages.

CherryPy & Flask are also popular web frameworks.

For scraping data from websites, the library **BeautifulSoup** is very useful, and leads to better results than building your own scrapper with regular expressions.

While Python does offer modules for programmatically accessing websites, such as "urllib", they are quite cumbersome to use. Third-party library requests make it much easier to use HTTP requests.

A number of third-party modules are available that make it much easier to carry out scientific and mathematical computing with Python.

The module **matplotlib** allows you to create graphs based on data in Python.

The module **NumPy** allows for the use of multidimensional arrays that are much faster than the native python solution of nested lists. It also contains functions to perform mathematical operations such as matrix transformations on the arrays.

The library **SciPy** contains numerous extensions to th functionality of NumPy.

Python can also be used for **game development**. Usually, it is used as a scripting language for games written in other languages, but it can be used to make games by itself.

Boolean Logic[edit]

Two Boolean values: True & False

Can be compared using the equal to operator == (Comparison Operator), not equal operator (!=), < or >, >= or <=

Greater than & smaller than operators can also be used to compare strings lexicographically (The alphabetical order of words is based on the alphabetical order of their component letters).

Boolean logic is used to make more complicated conditions for if statements that rely on more than one condition.

Python's Boolean operators are "**and**", "**or**" & "**not**". Python **uses words** for its Boolean operators.

Break[edit]

To end a loop pre-maturely, the break statement can be used. When encountered in a loop, the break statement causes the loop to finish immediately.

Using the break statement OUTSIDE THE LOOP CAUSES AN ERROR.

Continue[edit]

Unlike break, continue jumps back to the top of the loop, rather than stopping it.

Basically, the continue statement stops the current iteration & continues with the next one.

Using the continue statement OUTSIDE THE LOOP CAUSES AN ERROR.

List Slices[edit]

List slices provide a more advanced way of retrieving values from a list. Basic list slicing involves **indexing a list with two colon separated integers**. This **returns a new list** containing all the values in the old list between the specified indices.

>>>squares = [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

>>>print(squares[2:6])

Output: [4, 9, 16, 25]

Like the arguments to range, the first index provided in a slice is included in the result but the second isn't. If the first number in a slice is omitted, it is taken to be the start of list. If the second number is omitted, it is taken to be the end of the list.

Slicing can also be done on tuples.

List slices can also have a third argument, representing the step, to include only alternate value sin the slice.

Negative values can be used in list slicing as well as normal indexing. When negative values are used for the first & second values in a slice, they count from the end of the list.

If the negative value is used for the step, the slice is done backwards. using [::-1] is common & idiomatic way to REVERSE a list.

List Comprehensions[edit]

List comprehensions are a useful way of quickly creating lists whose contents obey a simple rule.

>>>cubes = [i3 for i in range(5)]**

>>>print(cubes)

List comprehensions are **inspired by set-builder notation** in mathematics.

A list comprehension can also contain an if statement to enforce a condition on values in the list.

>>>evens = [i2 for i in range(10) if i**2 % 2 == 0]**

>>>print(evens)

Output: [0, 4, 16, 36, 64]

Trying to create a list in a very extensive range will result in a MemoryError. This issue is solved by "generators".

Map[edit]

The built-in function map is a very useful higher order function that operates on lists (or similar objects called iterables)

The function map takes a function & an iterable as arguments, and returns a new iterable with a function applied to each argument.

```
>>>def add(x):
    return(x+5)

>>>nums = [11, 22, 33, 44, 55]

>>>result = list(map(add, nums))

>>>print(result)
```

Output: [16, 27, 38, 49, 60]

Useful String Functions[edit]

join -- Joins a list of strings with another string as a separation.

replace -- Replaces one substring in a string with another.

startswith & **endswith** -- Determines if there is a substring at the start and end of a string respectively.

To change the case of a string we use **lower()** & **upper()**

The method "split" is the opposite of join, turning a string with a certain separator into a list.

```
>>> print(".".join(["hello","world"]))

--> Output: hello.world
```

```
>>> print("hello me".replace("me", "world"))
```

--> Output: hello world

```
>>> print("This world".startswith("This"))
```

--> Output: True

```
>>> print("hello".upper())
```

--> Output: HELLO

```
>>> print("spam", "hello", "world")
```

--> Output: spam hello world

#Top

Recursion[edit]

The fundamental part of recursion is self reference -- functions calling itself.

The base case acts as the exit condition of the recursion.

Recursion can also be indirect. One function can call a second, which calls the first, which calls the second and so on... This can occur with any number of functions.

```
>>>def is_even(x):
        if x == 0:
              return(True)
        else:
              return(is_odd(x-1))
>>>def is_odd(x):
        return(not is_even(x))
```

>>>print(is_odd(17)) --> True

>>>print(is_even(23)) --> False

To find the factorial of a number, recursion technique is used extensively

```
>>>def factorial(n):
    if n == 1:
        return 1
    return n*factorial(n-1)
```

>>>print (factorial(5)) --> 120

Uses of sep/end in print function[edit]

The print() function prints the specified message to the screen, or other standard output device. The message can be a string, or any other object, the object will be converted into a string before written to the screen.

The function "sep" stands for separation. It is used to 'separate' multiple values of print statement with the provided argument

1.Example:

>>> print("Hello", "how are you?", sep = ",")

>>> Output : Hello, how are you?

2.Example: >>> x = "apple"

>>> y = "banana"

>>> z = "cherry"

>>> print(x,y,z, sep = ", ")

>>> Output : apple, banana, cherry

The function "end" specifies what to print at the end. By default a new line is inserted at the end.

1.Example:

>>> print("Python is fun", end = ".")

>>> Output : Python is fun.

2.Example:

>>> print("Hello", "how are you", sep = ", ",end = "?")

>>>Output: Hello, how are you?

Lecture Notes:Week 5

From JOCWiki

Contents

[hide]

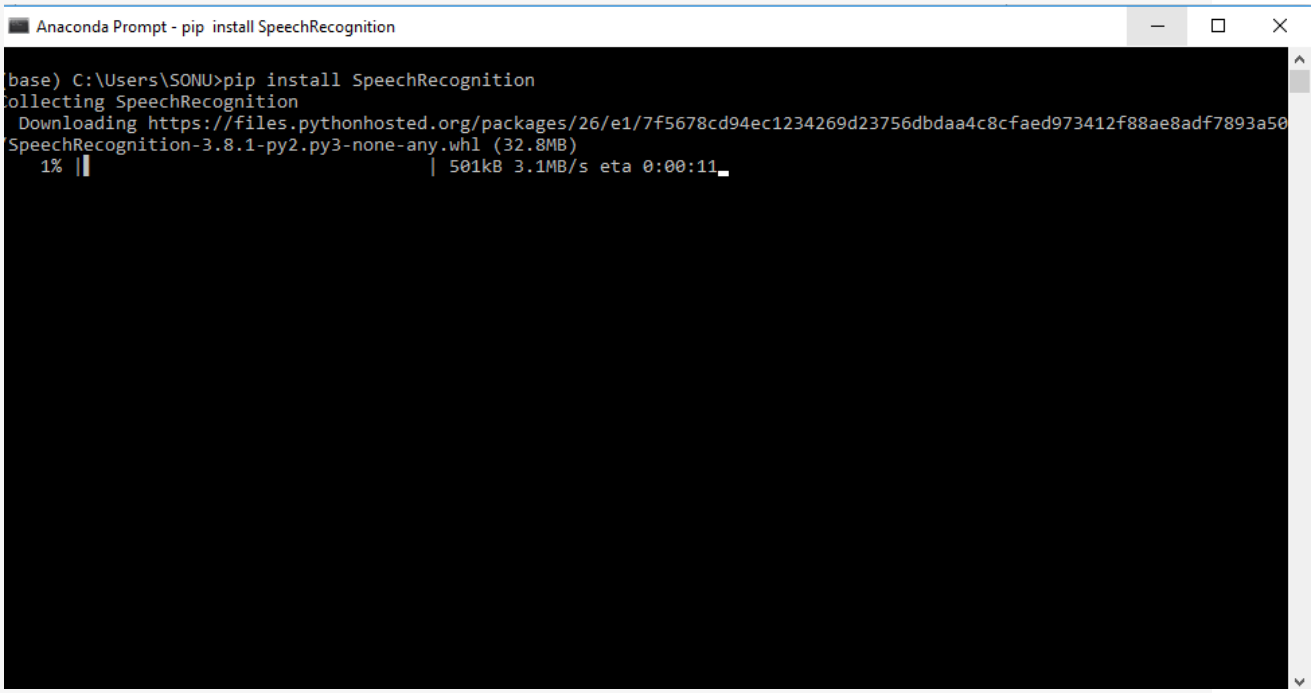
- 1 How to install Speech Recognition module using Anaconda Prompt
  - 1.1 Anaconda official website steps to install "speech recognition"
- 2 PROGRAM TO CONVERT SPEECH TO TEXT
- 3 MONTE HALL PROGRAM
  - 3.1 Guessing the right cup
    - 3.1.1 PROGRAM 1 for the above game:
    - 3.1.2 PROGRAM 2 for the above game:
  - 3.2 EXPLANATION of MONTE HALL PROGRAM
    - 3.2.1 Break & Continue in Python
- 4 MONTE HALL PROGRAM WITH K DOORS
  - 4.1 Code
  - 4.2 Outputs and Explanations
- 5 MONTE HALL PROGRAM WITH X GOAT DOORS AND Y BMW DOORS
  - 5.1 Code
  - 5.2 Outputs and Explanations
- 6 CHEAT PROOF: Rock Paper Scissor Game
  - 6.1 getpass() Library in Python
- 7 PROGRAM for different sorting techniques on a list :
  - 7.1 QUICK SORT
  - 7.2 INSERTION SORT
    - 7.2.1 An illustrative example
  - 7.3 MERGE SORT
    - 7.3.1 How function calls happen in MergeSort
  - 7.4 BUBBLE SORT
  - 7.5 SORTING BY SWAPING RANDOM INDEXES
- 8 BINARY SEARCH
- 9 DICTIONARIES IN PYTHON
  - 9.1 Creating a Dictionary
  - 9.2 Accessing objects from a dictionary
  - 9.3 Reassigning values associated with keys
  - 9.4 Methods available in dictionary
- 10 TUPLES IN PYTHON
  - 10.1 Constructing Tuples
  - 10.2 Basic Tuple Methods
  - 10.3 Immutability
- 11 LAMBDA FUNCTION IN PYTHON
  - 11.1 Examples
  - 11.2 Why Use Lambda Functions?
- 12 Keys to Dictionary
- 13 Dictionary Functions
- 14 Common Exceptions
- 15 Exception Handling
- 16 "finally"
- 17 Raising exceptions
- 18 Tuples (Additional Information)
- 19 When to use a Dictionary ?

- 20_When to use Other Types ?
- 21_Ternary Operator
- 22_Assertions
- 23_Text Analyzer Program (Involves working with files)
- 24_Python Enhancement Proposals (PEP)
- 25_PEP 8 (Suggestions)
- 26_Three Doors and a Twist Monte hall

How to install Speech Recognition module using Anaconda Prompt[edit]

You might be struggling while installing Speech Recognition module using command prompt/terminal. To install it in an easier way follow the below given steps:

1. Open Anaconda Prompt.
2. Type pip install SpeechRecognition *//Make sure you have an active internet connection*
3. Module will be installed successfully.



Anaconda official website steps to install "speech recognition"[edit]

Posted By: Arunkumar | Doubts on this topic?

If you are using Anaconda and still the above process didn't work for you! Try using this official Anaconda Website link to install the package. I am running Anaconda on Linux(Ubuntu 18).

anaconda.org/conda-forge/speechrecognition

Installation Steps.

To install this package with conda run one of the following:

```
conda install -c conda-forge speechrecognition
```

or

```
conda install -c conda-forge/label/cf201901 speechrecognition
```

```
File  Edit  View  Search  Terminal  Help
[desktop:~]$ conda install -c conda-forge speechrecognition
Collecting package metadata: done
Solving environment: done

## Package Plan ##

  environment location: /home/.../anaconda3

  added / updated specs:
    - speechrecognition


The following NEW packages will be INSTALLED:

  speechrecognition  conda-forge/linux-64::speechrecognition-3.6.3-py37_1000


Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
[desktop:~]$
```

PROGRAM TO CONVERT SPEECH TO TEXT[edit]

```
import speech_recognition as sr

AUDIO_FILE = ("Sampleaudio.wav")

r = sr.Recognizer()

with sr.AudioFile(AUDIO_FILE) as source:
    audio = r.record(source)

try :
```

```
    print("The audio file contains: " + r.recognize_google(audio))
except sr.UnknownValueError:
    print("Google Speech Recognition could not understand audio")
except sr.RequestError:
    print("Could not request results from Google Speech Recognition")
```

MONTE HALL PROGRAM[edit]

Posted By: Som

Guessing the right cup[edit]

In this game there are three cups and under one cup there's a coin and the player has to guess under which cup the coin resides.

Step 1: The player is asked to guess a cup out of the three cups.

Step 2: Now among the two remaining cups, either both the cups are empty (if the player's guess is correct) or one cup is empty (if the player's guess is incorrect). In either of the situation there's at least one cup which is empty. So, the empty cup is showed to the player.

Step 3: The player is now asked if he/she wants to continue with his/her original guess or wants to swap his/her answer.

Step4 : After the player has decided what to do, the player is shown the cup under which the coin resides.

PROGRAM 1 for the above game:[edit]

```
 1 import random
 2 cup = ['_','_','_']
 3 display = [1,2,3]
 4 print(display)
 5
 6 ind_r = random.randint(0,2)
 7 cup[ind_r] = 'GOLD'
 8 ind_ch = int(input('Guess the position of GOLD'))-1
 9
10 if 0 != ind_r and 0 != ind_ch:
11     ind_show = 0
12 elif 1 != ind_r and 1 != ind_ch:
13     ind_show = 1
```

```
elif 2 != ind_r and 2 != ind_ch:
    ind_show = 2

if ind_ch != ind_r:
    display[ind_show] = cup[ind_show]
    print(display)

    ask = int(input('Press 0 to swap and 1 to continue'))
    if ask == 1:
        print(cup)
        print('Sorry you lose')
    else:
        print(cup)
        print('You Win')

else:
    display[ind_show] = cup[ind_show]
    print(display)

    ask = int(input('Press 0 to swap and 1 to continue'))
    if ask == 1:
        print(cup)
        print('You Win')
    else:
        print(cup)
        print('Sorry you lose')
```

PROGRAM 2 for the above game:[edit]

Posted By: Som

```
import random
n= random.randint(1,3)
ch=int(input())
if ch==n:
    if n==1:
        r=random.choice([2,3])
```

```
 7
 8         print(" Jar" , r , "is empty")
 9         if r==2:
10             t=3
11         else:
12             t=2
13     elif(n==2):
14         r=random.choice([1,3])
15
16         print("Jar" , r , "is empty")
17         if r==1:
18             t=3
19         else:
20             t=1
21     else:
22         r=random.choice([1,2])
23         print("Jar" , r , "is empty")
24         if r==1:
25             t=2
26         else:
27             t=1
28 else:
29     if n==1:
30         if ch==2:
31             print("Jar 3 is empty")
32             t=1
33         else:
34             print("Jar 2 is empty")
35             t=1
36     elif n==2:
37         if ch==1:
38             print("Jar 3 is empty")
39             t=2
40         else:
41             print("Jar 1 is empty")
42             t=2
```

```
    else:
        if ch==1:
            print("Jar 2 is empty")
            t=3
        else:
            print("Jar 1 is empty")
            t=3
s=int(input("Press 1 to swap and 0 to stay"))
if(s==1):
    ch=t
print("Jar" , n , "has coin")
if(ch==n):
    print("Win")
else:
    print("Lose")
```

EXPLANATION of MONTE HALL PROGRAM[edit]

Posted By: Som

```
import random
doors = [0]*3
goatdoor = []
swap=0 #number of swap wins
dont_swap=0 #number of dont swap wins
j = 0
x=random.randint(0,2) #x door will comprise of BMW
doors[x]="BMW"
while j<10:
    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) #opens a door that comprises of a
goat
   while(door_open==choice): #door_open shouldn't be equal to choice

       door_open=random.choice(goatdoor)
    ch=input("Do you want to swap? y/n")
    if(ch=="y"):
        if(doors[choice]=="Goat"):
            print("Player Wins")
            swap+=1 #increments swap variable with 1
        else:
            print("Player lost")
    else:
        if(doors[choice]=="Goat"):
            print("Player Lost")
        else:
            print("Player Wins")
            dont_swap+=1 #increments dont_swap variable with 1

   j+=1
```

print('No. of swap wins' ,swap)

print('No. of dont swap wins' , dont_swap)

Explanations

- *Yellow Shaded Region :*

  In place of the shaded region, we could also you the following code:

```
    for i in range(0,3):
        if (i != x):
            doors[i] == "Goat"
            goatdoor.append(i)
```

  OR

```
    for i in range(0,3):
        if (i==x):
            continue
        doors[i]="Goat"
        goatdoor.append(i)

WHY ACTUALLY WE DONT NEED AN ELSE STATEMENT ?
```

Break & Continue in Python[edit]

Break: *Breaks out of the current closest enclosing loop.*
Continue: *Goes to the top of the closest enclosing loop.*

```
my_list=[1,2,3,4,5,6,7,8]
for item in my_list:
    if item == 5:
        continue
    print(item,end=' ')

OUTPUT : 1 2 3 4 6 7 8
```

Skips only 5

```
my_list=[1,2,3,4,5,6,7,8]
for item in my_list:
    if item == 5:
        break
    print(item,end=' ')

OUTPUT : 1 2 3 4
```

Skips everything from 5 onwards

```
for i in 'abcde':
    for j in range(6):
        if j == 2:
            continue
        print(f'{i}{j}',end=' ')
```

```
OUTPUT : a0 a1 a3 a4 a5 b0 b1 b3 b4 b5 c0 c1 c3 c4 c5 d0 d1
d3 d4 d5 e0 e1 e3 e4 e5
```

Skips only 'j' value equal to 2

```
for i in 'abcde':
    for j in range(6):
        if j == 2:
            break
        print(f'{i}{j}',end=' ')
```

```
OUTPUT : a0 a1 b0 b1 c0 c1 d0 d1 e0 e1
```

Skips every 'j' value from 2 onwards

- *Violet Shaded Region :*
  - The outer IF loop checks whether the player has chosen to swap or not.
    - The inner IF loop checks if the choice made by the player was actually wrong (i.e the door had a goat inside it). So in this situation player has chosen to swap his decision which was actually wrong, therefore the player wins.
    - Else, now here the choice made by the player was correct (i.e not a goat door). So in this situation player has chosen to swap his decision which was actually correct, therefore the player loses.
  - The outer ELSE loop is executed when the player hasn't chosen to swap.
    - The inner IF loop checks if the choice made by the player was actually wrong (i.e the door had a goat inside it). So in this situation player has chosen to not to swap his decision which was actually wrong, therefore the player loses.
    - Else, now here the choice made by the player was correct (i.e not a goat door). So in this situation player has chosen not to swap his decision which was actually correct, therefore the player wins.

MONTE HALL PROGRAM WITH K DOORS[edit]

Posted By: Som

In Monte Hall problem, we have 3 doors. 2 of them comprise of goats and one of them comprises of BMW. Implement Monte Hall problem in python if we have k doors in which k-1 doors comprise of goats and 1 door comprises of BMW.

Code[edit]

```
1 import random
2 k=int(input("Enter the number of Doors ")) #stores the total number
doors provided by the user
3 choice=int(input(f"Enter your Door Number 0/1...{k-1} "))
```

```
 4 doors=[0]*k   #Creating a list with k elements with all the elements as 0
 5 bmw=random.randint(0,k-1)  #Generating a random door that will contain BMW
 6 doors[bmw]="BMW"
 7 goatdoor=[] #creating an empty list that will keep track of all the indices of goat doors
 8 for i in range(k):
 9     if(i==bmw):
10         continue
11     doors[i]="Goat"
12     goatdoor.append(i) #append the index of goat door in goatdoor list
13 while k>2:
14     door_open=random.choice(goatdoor)
15     while door_open == choice:   #Since we dont want the door chosen by the user to be opened
16         door_open=random.choice(goatdoor)
17     print(f"Door {door_open} has a Goat!")
18     goatdoor.remove(door_open) #remove the door that has already been opened
19     change=input("Do you want to change your door number? y/n ")
20     if(change=='y'): #Checks user input
21         choice=int(input("Enter your Door Number: "))   #Asking the user to enter another door number
22     k-=1
23
24 if(choice==bmw):
25     print("Hurrah! You won the BMW")
26 else:
27     print("Oops, You only have the goat to take :( ")
28     print(f"Actually Door {bmw} had the BMW") #display the door that had BMW in it.
```

Outputs and Explanations[edit]

Procedure

- First of all the user is asked the total number of doors.
- Then the user is asked his choice amongst the k doors
- Then one of the k doors is opened except the door chosen by the user and the door that has BMW
- Now the user is asked whether he wants to change his opinion or wants to stick to his choice.
- If he choses yes, then the new door number is asked. If he choses no then his choice remains the same.
- Then one amongst the remaining doors is opened.
- This keeps repeating until two doors are left.
- In his last choice if the door chosen by him contains BMW, then he gets BMW. Otherwise he gets a Goat.

Output 1

```
Enter the number of Doors 5

Enter your Door Number 0/1...4 3
Door 1 has a Goat!

Do you want to change the door? y/n n
Door 0 has a Goat!

Do you want to change the door? y/n n
Door 2 has a Goat!

Do you want to change the door? y/n y

Enter your Door Number: 4
Oops, You only have the goat to take :(
Actually Door 3 had the BMW
```

Explanation

- The total number of doors chosen by the the user is 5.
- User selects the 4th door.(i.e the door with number 3 on it).
- Door 1 has goat is displayed.
- User choses not to change his decision.
- Door 0 has goat is displayed.
- User choses not to change his decision.
- Door 2 has goat is displayed.
- User then choses to change his decision.
- His new door number is 4.
- A message is displayed stating that he didn't get the BMW, because the car was in door number 3.

Output 2

```
Enter the number of Doors 3

Enter your Door Number 0/1...2 0
Door 1 has a Goat!

Do you want to change the door? y/n y

Enter your Door Number: 2
Hurrah! You won the BMW
```

Explanation

- The total number of doors chosen by the user is 3.
- The user selects the first door(i.e the door with number '0' on it).
- Door 1 has goat is displayed.

- The user wishes to change his decision.
- The new door number is 2.
- A message is displayed stating that he wins BMW.

MONTE HALL PROGRAM WITH X GOAT DOORS AND Y BMW DOORS[edit]

Posted By: Som

In another variation of Monte Hall problem, we have x doors. Out of x doors, y doors comprise of goats and z comprise of BMWs. Implementing this variation of Monte Hall problem.

Code[edit]

```
import random
#stores the total number of doors provided by user
k=int(input("Enter the number of Doors: "))
#randomly generating 'nbmw' no of doors with BMW
nbmw=random.randint(1,int(k/2))
#displays the total number of BMW doors generated randomly
print(f"There are {nbmw} BMW doors! Good Luck")
#accepts the user's selected door number
choice=int(input(f"Enter your Door Number 0/1...{k-1}: "))
doors=[0]*k #creating a list doors of length k with all elements as 0
for i in range(nbmw):
    #while(1), means the condition for the while loop is true, this generates a infinite loop
    while(1):
        bmw=random.randint(0,k-1)
        #this only changes the indices where 0 is present
        if(doors[bmw] == 0):
            doors[bmw] = 'BMW'
            #since we have used a break statement here the loop doesn't trap in an infinite loop
            break
#creating a list that will keep a track of index of all goat doors
goatdoor=[]
for i in range(k):
```

```
    #if the index i of doors list already contains an elements BMW we dont change it
    if(doors[i]=="BMW"):
        continue
    doors[i]="Goat"
    #all the indexes containing goats are appended
    goatdoor.append(i)

#the players only gets to select doors until the no of goat doors is greater than BMW doors
while len(goatdoor)>nbmw:
    door_open=random.choice(goatdoor)
    #we keep generating a new door_open until the value of door_open is equal to choice
    while door_open == choice:
        door_open=random.choice(goatdoor)
    print(f"Door {door_open} has a Goat!")
    #remove the door that has already been opened.
    goatdoor.remove(door_open)
    change=input("Do you want to change the door? y/n: ")
    if(change=='y'):
        choice=int(input("Enter your Door Number: "))

#if the value chosen by the user is equal to the door containing BMW,
#then player wins a BMW.
if(doors[choice]=='BMW'):
    print("Hurrah! You won the BMW")
else:
    print("Oops, You only have the goat to take :( ")
    for i in range(len(doors)):
        if doors[i] == 'BMW':
            #displays all the doors containing BMW
            print(f'Door {i} had a BMW')
```

Outputs and Explanations[edit]

Procedure

- At first the user is asked the total number of doors
- Then randomly no of doors containing BMW are generated, with a condition that no of BMW doors lies between 1 and half of the total number of goat doors.
- The the total number of BMW doors are displayed (without mentioning their positions).
- Then the users choice is asked.
- After that one door is opened randomly, which is not amongst the BMW doors or the user's choice.
- Now the user is asked whether he wishes to change his decision or continue with the previous choice.
- If yes, then the new door number is asked.
- Then again a new door is opened, which is neither the one already opened nor a BMW door or the user's choice.
- Now, this keeps repeating until number of unopened goat doors is greater than number of BMW doors.
- Then if the last choice of user is equal to any of the BMW doors he wins a BMW otherwise he wins a goat.
- if he wins a goat, all the doors that contained BMW are revealed.

Examples:

- **Output 1:**

```
Enter the number of Doors: 8
There are 2 BMW doors! Good Luck
Enter your Door Number 0/1...7: 7
Door 0 has a Goat!
Do you want to change the door? y/n: n
Door 3 has a Goat!
Do you want to change the door? y/n: y
Enter your Door Number: 1
Door 4 has a Goat!
Do you want to change the door? y/n: y
Enter your Door Number: 2
Door 6 has a Goat!
Do you want to change the door? y/n: n
Hurrah! You won the BMW
```

Explanation

  - The total number of doors entered by the user is 8.
  - The total number of BMW doors(generated randomly) is shown which is equal to 2.
  - The user is asked about his choice of door number, which is 7.
  - Door 0 has a goat is displayed.
  - The user is now asked whether or not he wants to change his decision.
  - The users sticks to his choice.
  - Door 3 has a goat is displayed.
  - The user is asked whether he wants to change his decision, and the user switches to door number 1.
  - Door 4 has a goat is displayed.
  - The user is asked whether he wants to change his decision, and the user switches to door number 2.
  - Door 6 has a goat is displayed.
  - The user now stays with his decision of door number 2.
  - A message containing he has won a BMW is displayed.
- **Output 2**

```
Enter the number of Doors: 4
There are 2 BMW doors! Good Luck
Enter your Door Number 0/1...3: 0
Oops, You only have the goat to take :(
Door 1 had a BMW
Door 2 had a BMW
```

Explanation

- The total number of doors is asked, which is specified to 4 by the user.
- The total number of BMW doors(generated randomly) is shown which is equal to 2.
- The user is asked about his choice of door number, which is given as '0' (i.e first door) by the user.
- Total number of unopened goat doors is equal to the total number of BMW doors so the conclusion is shown.
- A message stating that the user has lost and only won a goat is shown.
- All the door numbers that had BMW are shown i.e door 1 and door 2 in this case.

CHEAT PROOF: Rock Paper Scissor Game[edit]

Posted By: Som

Do you think the rock, paper, scissor game that we saw in the screen cast was actually cheat proof ?

Heres a cheat proof version of the same:

```
plist = ['rock','paper','scissor']
for i in range(len(plist)):
    print(plist[i],end='  ')
print()
for i in range(len(plist)):
    print(f'({i+1})',end='    ')
print()
p1point = 0
p2point = 0
play = True #Note The T has to be in uppercase
while(play):
    p1choose = int(getpass.getpass(prompt='PLAYER 1: Enter your choice'))
    p1choose -= 1 #So that the index of array is correct i.e 0,1 or 2
```

```
14      p2choose = int(getpass.getpass(prompt='PLAYER 2: Enter your
choice'))
15      p2choose -= 1
16      if (plist[p1choose] == plist[p2choose]):
17          print("ITS A DRAW!")
18      else:
19          if(plist[p1choose]=='rock' and plist[p2choose]=='scissor'):
20              p1point += 1
21              print("PLAYER 1 WON")
22          elif(plist[p1choose]=='rock' and plist[p2choose]=='paper'):
23              p2point += 1
24              print("PLAYER 2 WON")
25          elif(plist[p1choose]=='scissor' and
plist[p2choose]=='paper'):
26              p1point += 1
27              print("PLAYER 1 WON")
28          elif(plist[p1choose]=='scissor' and
plist[p2choose]=='rock'):
29              p2point += 1
30              print("PLAYER 2 WON")
31          elif(plist[p1choose]=='paper' and plist[p2choose]=='rock'):
32              p1point += 1
33              print("PLAYER 1 WON")
34          elif(plist[p1choose]=='paper' and
plist[p2choose]=='scissor'):
35              p2point += 1
36              print("PLAYER 2 WON")
37      p1play = input('PLAYER 1, DO YOU WISH TO CONTINUE Y/N').lower()
38      if p1play == 'y':
39          p2play = input('PLAYER 2, DO YOU WISH TO CONTINUE
Y/N').lower()
40          if p2play == 'y':
41              pass
42          else:
43              play = False
44      else:
```

```
45          play = False #Note The F has to be in uppercase
46 #After exiting the while loop
47 print(f"PLAYER 1 POINTS = {p1point}")
48 print(f"PLAYER 2 POINTS = {p2point}")
49 if p1point > p2point:
50     print("PLAYER 1 wins")
51 elif p1point == p2point:
52     print("DRAW !")
53 else:
54     print("PLAYER 2 Wins")
```

OUTPUTS

```
rock  paper  scissor
(1)    (2)    (3)
PLAYER 1: Enter your choice••••••••

PLAYER 2: Enter your choice •
```

```
rock  paper  scissor
(1)    (2)    (3)
PLAYER 1: Enter your choice••••••••
PLAYER 2: Enter your choice••••••••
PLAYER 2 WON

PLAYER 1, DO YOU WISH TO CONTINUE Y/N
```

```
rock  paper  scissor
(1)    (2)    (3)
PLAYER 1: Enter your choice••••••••
PLAYER 2: Enter your choice••••••••
PLAYER 2 WON
PLAYER 1, DO YOU WISH TO CONTINUE Y/NN
PLAYER 1 POINTS = 0
PLAYER 2 POINTS = 1
PLAYER 2 Wins
```

Here we use the getpass() library in python, which helps us to keep our input secret.

getpass() Library in Python[edit]

An example to the show the usage of getpass library

```
1 i=0
2 while i<3:
```

```
    password = getpass.getpass(prompt='Your Favourite Movie').lower()
    if (password == 'avengers'):
        print("Welcome")
        break
    else:
        print("Sorry that's incorrect")
        i += 1
else:
    print("You have exceeded the number of attempts")
```

NOTE : The usage of while else statement in python

PROGRAM for different sorting techniques on a list :[edit]

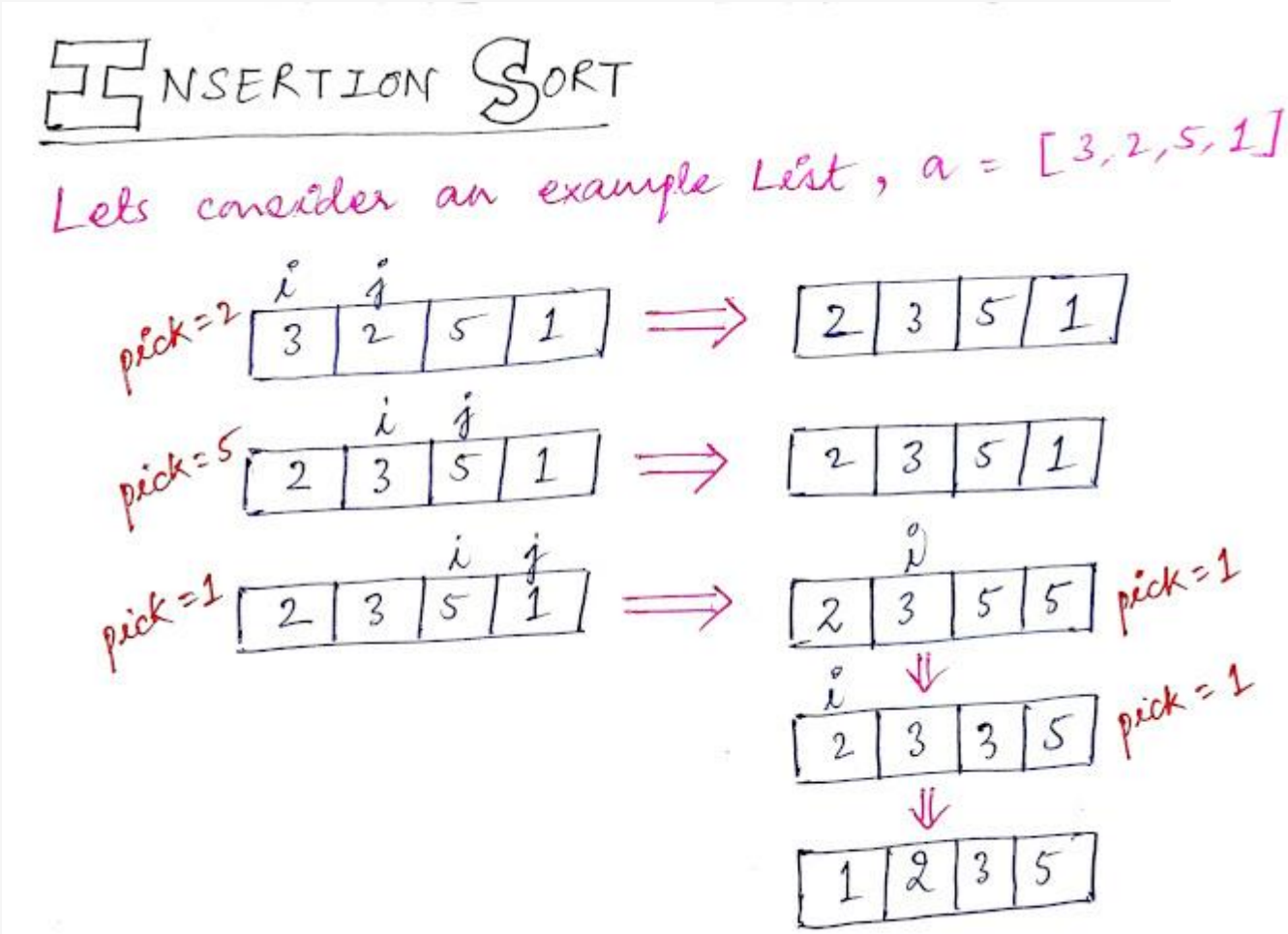
QUICK SORT[edit]

```
def partition(a,p,r): #a is the list ,p is the first index & r is the last index
    x = a[r] #x is the pivot element
    i = p-1

#Pivot element is placed such that all elements to left of pivot are smaller than it
#and all elements to the right of pivot are greater than it

    for j in range(p,r):
        if a[j] <= x:
            i += 1
            a[i],a[j] = a[j],a[i]
    a[i+1],a[r] = a[r],a[i+1]
    return i+1

def quicksort(a,p,r):
    if p < r:
        q = partition(a,p,r)
        quicksort(a,p,q-1) #Sorting LHS of qth element
        quicksort(a,q+1,r) #Sorting RHS of qth element
```

INSERTION SORT[edit]

```
def insertion_sort(a):
    for j in range(1,len(a)):
        pick = a[j] #Picking elements from the list
        i = j-1
#Placing the picked element on correct position
        while (i >= 0 and a[i]>pick):
            a[i+1] = a[i]
            i -= 1
        a[i+1] = pick
#All elements to the left of picked element are sorted at this point
```

An illustrative example[edit]



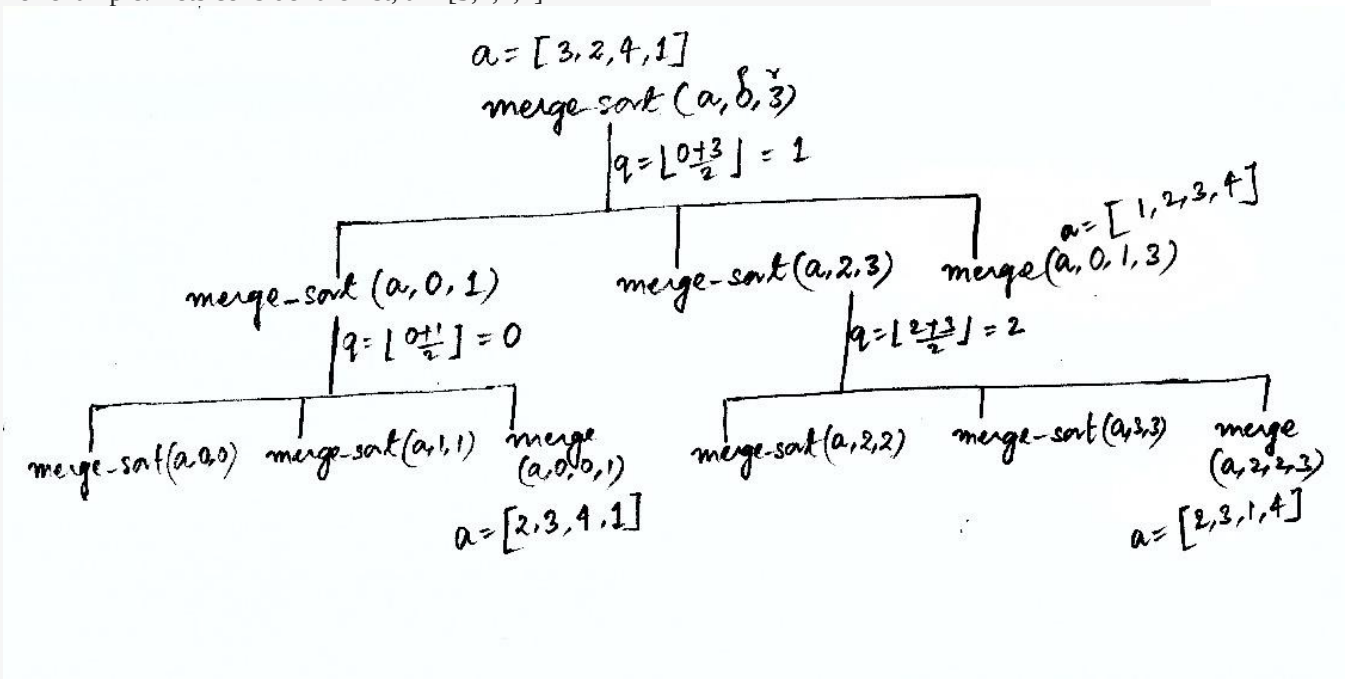
MERGE SORT[edit]

```
# Merging two sorted arrays
# First subarray is a[p to q]
# Second subarray is a[q+1 to r]
def merge(a,p,q,r):
    l = q-p+1
    r = r-q

# create temp arrays
    la = [0]*l; ra = [0]*r
# copying data to these temp arrays
    for i in range(l):
        la[i] = a[p+i]
    for j in range(r):
        ra[j] = a[q+1+j]

    i = 0 #Initial index of first subarray
    j =0  #Initial index of second subarray
    k = p #Initial index of merged subarray

#Merging Temp arrays back to a[p to r]
    while i < l and j < r :
        if la[i] <= ra[j]:
            a[k] = la[i]
            i += 1
        else:
            a[k] = ra[j]
            j += 1
        k += 1

# Copy the remaining elements of L[], if any
    while i < l:
        a[k] = la[i]
        i += 1
        k += 1

```

```
36  # Copy the remaining elements of R[], if any
37      while j < r:
38          a[k] = ra[j]
39          j += 1
40          k += 1
41
42  def merge_sort(a,p,r):
43      if p < r:
44          q = int((p+r)/2)
45          merge_sort(a,p,q)
46          merge_sort(a,q+1,r)
47          merge(a,p,q,r)
```

How function calls happen in MergeSort[edit]

For example: Lets consider the list, a = [3,2,4,1]



- merge(a,0,0,1) is executed firstly and after this our list 'a' becomes a = [2,3,4,1]
- merge(a,2,2,3) is executed after this and then our list 'a' becomes a = [2,3,1,4]
- merge(a,0,1,3) is executed finally and then our list 'a' becomes a = [1,2,3,4]

BUBBLE SORT[edit]

Sorting in Ascending order

```
1 def bubble(a):
```

```
    n = len(a)
    for i in range(n):
        for j in range(n-i-1):
            if a[j] >= a[j+1]:
                a[j],a[j+1] = a[j+1],a[j]
```

OUTPUT

```
n = int(input("Enter the no of elements in array "))
list1 = []
for i in range(n):
    list1.append(int(input(f"Enter the element {i+1} ")))
bubble(list1)
print("Sorted Array (Ascending Order) :",list1)
```

```
Enter the no of elements in array 4
Enter the element 1 3
Enter the element 2 5
Enter the element 3 1
Enter the element 4 9
Sorted Array (Ascending Order) : [1, 3, 5, 9]
```

Sorting in Descending order

```
def bubble(a):
    n = len(a)
    for i in range(n):
        for j in range(n-i-1):
            if a[j] <= a[j+1]:
                a[j],a[j+1] = a[j+1],a[j]
```

OUTPUT

```
n = int(input("Enter the no of elements in array "))
list1 = []
for i in range(n):
    list1.append(int(input(f"Enter the element {i+1} ")))
bubble(list1)
print("Sorted Array (Descending Order) :",list1)
```

```
Enter the no of elements in array 4
Enter the element 1 5
Enter the element 2 1
Enter the element 3 3
Enter the element 4 2
Sorted Array (Descending Order) : [5, 3, 2, 1]
```

NOTE: The swapping technique used above

Conventional Way

```
swap = a[j]
a[j] = a[j+1]
a[j+1] = swap
```

A smarter approach that can be adopted in python

```
a[j],a[j+1] = a[j+1],a[j]
 Swaps a[j] and a[j+1]
```

SORTING BY SWAPING RANDOM INDEXES[edit]

```
from random import randint
list_1 =[]
swap = 0
do = True
n = int(input())
for i in range(n):
    list_1.append(int(input()))
while (do) :
```

```
    check = 0
    for i in range(n-1):
        if list_1[i] <= list_1[i+1]:
            check += 1
        else:
            break
    if check == n-1:
        do = False
```

```
    else:
        i = randint(0,n-1)
        j = randint(0,n-1)
        swap = list_1[i]
        list_1[i] = list_1[j]
        list_1[j] = swap

for i in range(n-1):
    print(list_1[i], end=" ")
print(list_1[n-1],end="")
```

Note: The highlighted part checks if the list is sorted or not

Below is a program to test the working of the above code

```
from random import randint
list_1 =[]
lsort = []
swap = 0
turn = 0
do = True
n = int(input())
for i in range(n):
    list_1.append(int(input()))
for i in list_1:
    lsort.append(i)
```

```
lsort.sort()

while (do) :
    check = 0
    for i in range(n-1):
        if list_1[i] <= list_1[i+1]:
            check += 1
        else:
            break
    if check == n-1:
        do = False
    else:
        turn += 1
        print(f'Turn {turn}')

        i = randint(0,n-1)
        print(f'i = {i}')

        j = randint(0,n-1)
        print(f'j = {j}')

        swap = list_1[i]
        list_1[i] = list_1[j]
        list_1[j] = swap

for i in range(n-1):
    print(list_1[i], end=" ")

print(list_1[n-1],end="")<hr>
```

Turn Variable has been added to see what indexes are swapped after each turn

An Example Below

```
4
4
3
1
2
Turn 1
i = 2
j = 0

Turn 2
i = 3
j = 3

Turn 3
i = 3
j = 0

Turn 4
i = 1
j = 2

Turn 5
i = 1
j = 0

Turn 6
i = 3
j = 0

1 2 3 4
```

```
# INPUT LIST : [4,3,1,2]
# At Turn 1 : Indexes 2 and 0 are swapped.
                Updated List : [1,3,4,2]
```

```
    # At Turn 2 : Indexes 3 and 3 are swapped.
                    Updated List : [1,3,4,2]
    # At Turn 3 : Indexes 3 and 0 are swapped.
                    Updated List : [2,3,4,1]
    # At Turn 4 : Indexes 1 and 2 are swapped.
                    Updated List : [2,4,3,1]
    # At Turn 5 : Indexes 1 and 0 are swapped.
                    Updated List : [4,2,3,1]
    # At Turn 6 : Indexes 3 and 0 are swapped.
                    Updated List : [1,2,3,4]
    LIST IS SORTED.
```

Note: You can avoid choosing same indexes as in turn 2 above, but for the sake of simplicity, it is avoided here

BINARY SEARCH[edit]

```
1  def binsr(arr,ele):
2      arr.sort()
3      n1=0
4      n2=len(arr)-1
5      while(n1<=n2):
6          mid=int((n1+n2)/2)
7          if ele > arr[mid]:
8              n1=mid+1
9
10         elif ele < arr[mid]:
11             n2=mid-1
12
13         else:
14             print("Element Found at pos",mid)
15             return
16
17     print("Element not Found!")
```

Note: The *FLAG* variable is actually not required, *RETURN* alone does the job.

- Binary search is better as compared to Linear Search.
- Time Complexity of binary search is O(log n) whereas Time Complexity of Linear Search is O(n).
- Binary Search after every iteration reduces the search space by half.

Another Program for Binary Search

```
def binary_search(arr,x):
    arr.sort()
    mid=int(len(arr)/2)
    while True:
        if x<arr[mid]:
            arr = arr[:mid] #Slicing out the lower half
        elif x>arr[mid]:
            arr = arr[mid:] #Slicing out the upper half
        else:
            print("Element Found")
            break
        if mid==0: #To exit the While Loop
            print("Element Not Found")
            break
        mid=int(len(arr)/2)
```

DICTIONARIES IN PYTHON[edit]

1. Dictionaries are unordered mappings for storing objects.
2. We saw how lists store values in an ordered fashion, but a dictionary uses key:value pairing instead stored in an unordered manner.
3. In Lists an item is referred using its index, but in a dictionary a value is fetched using its key. So, to fetch an item in a dictionary you don't actually need to know its position, only the key would do.
4. Dictionary uses curly braces and colons to separate the key:value pair and comma to separate the keys
5. The keys must be unique but the values can be redundant.
6. Unlike in a list, a dictionary cannot be sorted

- Dictionaries are data structures used to map arbitrary keys to values. Lists can be thought of as dictionaries with integer keys with a certain range.
- Dictionaries can be indexed in the same way as lists, using square brackets containing keys.
- Each element in a dictionary is represented by a **key : value** pair.
- Trying to index a key that isn't a part of the dictionary returns a **KeyError**.
- A dictionary can store any types of data as values. An empty dictionary is defined as {}.

DICTIONARIES
Dictionaries -one of the data types- are a sequence of heterogeneous key-value pairs confined in curly braces.

Properties:

- The keys must be explicit.
- Values can be repeated.
- Indexing gives the position of keys.

Creating a Dictionary[edit]

Creating an empty dictionary:

```
my_dic={}
```

Adding elements to a dictionary:

```
my_dic[0]="First_value"
my_dic[1]="Second_value"
```

Dictionary with string as keys

```
my_dic = {'key1':'value1','key2':'value2'}
```

Dictionary with integer as keys

```
my_dic = {1 :'value1',2:'value2'}
```

Dictionary with integer/string as keys

```
my_dic = {1 :'value1','k2':'value2','k3':'value3'}
```

Dictionary with integer as values

```
my_dic = {'books' :200,'pencils':50}
```

Lists within dictionary

```
my_dic = {1 :[1,2,3],2:'value2'}
```

Dictionary within dictionary

```
my_dic = {1 :{'k1':'v1','k2':'v2'},2:'value2'}
```

Accessing objects from a dictionary[edit]

Accessing Keys

```
my_dic = {1 :{'k1':'v1','k2':'v2'},2:'value2'}
for i in my_dic.keys():
    print(i)
```

Accessing Values

```
my_dic = {1 :{'k1':'v1','k2':'v2'},2:'value2'}
for i in my_dic.values():
    print(i)
```

Accessing Keys and Values

```
my_dic = {1 :'values',2:'value2'}
for i in my_dic.items():
    print(i)
```

Accessing elements via keys

```
my_dic = {1 :{'k1':'v1','k2':'v2'},2:'value2'}
print(my_dic[1]['k2'])
```

Deleting a key value pair

```
my_dic = {1 :'value1',2:'value2'}
del my_dic[1]
```

Reassigning values associated with keys[edit]

EXAMPLE 1

```
costs = {'mango':60,'apple':120,'guava':70,'grapes':40}
costs['apple']=150
```

EXAMPLE 2

```
costs =
{'fruits':{'mango':60,'apple':120,'guava':70,'grapes':40},'sta
tionery':{'pen':20,'pencil':30,'rubber':5}}
costs['fruits']['apple'] = 150
```

Methods available in dictionary

1. .has_key()
2. .clear()
3. .copy()
4. .keys()
5. .values()
6. .items()
7. .update()
8. .get()
9. .pop()
10. .fromkeys()

TUPLES IN PYTHON[edit]

- We have already seen lists and dictionaries in pyhton, now its time to get familier with tuples.
- In Python tuples are quite similar to lists, however, it is worth nothing unlike lists they are immutable i.e. they cannot be changed.
- One must use tuples to store things that shouldn't be changed, such as names of the month, or dates on a calendar.

Now we would discuss the following one by one :

```
1.) Constructing Tuples
2.) Basic Tuple Methods
3.) Immutability
4.) When to Use Tuples
```

You'll be curious of how to use tuples based on what you've learned about lists. We can treat them very similarly with the major distinction being that tuples are immutable.

Constructing Tuples[edit]

```
In [1]: #We used [] to create lists
        #We used {} to create dictionaries
        #Now We use () to create a tuple

In [10]: #Creating a tuple
         tup = ('a','b','c','d','e','f')

In [11]: #Checking the type of our variable 'tup'

         type(tup)

Out[11]: tuple

In [12]: #Checking the length of our variable 'tup'
         len(tup)  #just as we did in lists

Out[12]: 6

In [13]: #Indexing to fetch elements of the tuple

         tup[1] #Note we use square brackets here

Out[13]: 'b'

In [14]: #Slicing
         tup[1::2]

Out[14]: ('b', 'd', 'f')
```

Basic Tuple Methods[edit]

```
In [234]: tup = (2,3,4,4,1,7,1,6,1)

In [235]: # Returns the index of the value given
          tup.index(4)

Out[235]: 2

In [238]: #Returns the no. of occurences of the value given
          tup.count(3)

Out[238]: 1
```

Immutability[edit]

```
In [244]: tup = ('apple','mango','orange','grapes')

In [248]: tup[0] = 'guava'

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-248-d04f903ec13d> in <module>
----> 1 tup[0] = 'guava'

TypeError: 'tuple' object does not support item assignment


In [249]: tup.append('berries')

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-249-0ddf5cfe3703> in <module>
----> 1 tup.append('berries')

AttributeError: 'tuple' object has no attribute 'append'
```

- Immutability is an important aspect with tuples that creates a major distinction between Lists and Tuples.
- So once a tuple is created, you can not modify its elements.
- So tuples are basically created with an intention, so that its elements stay intact.
- For example,
  `tup = ('Monday','Tuesday,'Wednesday','Thursday','Friday','Saturday','Sunday')`
- Also one cannot append new elements to a tuple once it is created

LAMBDA FUNCTION IN PYTHON[edit]

The keyword def is used when we wish to define normal functions. The lambda keyword is used when we want to create anonymous functions. This function can take any number of arguments, but can only have one expression.

syntax

lambda arguments : expression

Examples[edit]

- ```
  square = lambda num: num*num
  print(square(int(input("Enter a number: "))))
  ```

  ```
  Enter a number: 15
  225
  ```

  In this the lambda function named square is made that accepts a number 'num' from the user and calculates the square of the number.

- ```
  product = lambda num1, num2, num3 : num1*num2*num3
  print(product(3,4,5))
  ```

  ```
  60
  ```

  In this the lambda function named product is made that accepts three numbers namely num1, num2, num3 and calculates the product of these three numbers.

Why Use Lambda Functions?[edit]

Lambda functions are very useful when we use them as anonymous functions inside another function. This can be easily understood by the following example:

```
def powerfunc(power):
    return lambda num: num ** power
```

```
square=powerfunc(2)
cube=powerfunc(3)
```

```
result1=square(6)
```

```
result2=cube(5)
```

```
print(result1)
```

```
36
```

```
print(result2)
```

```
125
```

Explanation

- First we define a function named powerfunc.
- We take the help of lambda function and define a anonymous function where num is a number that is raised to the power equal to 'power' variable.
- Now we define a 'square' function with the help of 'powerfunc' function by passing 2 to 'powerfubnc', where 2 acts as the value of 'power' variable.
- So the function 'square'(i.e function derived by 'powerfunc') can now be used as a function to find the square of a number 'num'.
- Now we define a 'cube' function with the help of 'powerfunc' function by passing 3 to 'powerfubnc', where 3 acts as the value of 'power' variable.
- So the function 'cube'(i.e function derived by 'powerfunc') can now be used as a function to find the cube of a number 'num'.
- Now, a new variable result1 is created that is assigned the value equal to 'square(6)'
  i.e value of num=6 and power=2.
- Then, a new variable result2 is created that is assigned the value equalt to 'cube(5)'
  i.e. value of num=5 and power=3.
- Value of result1(=36) is printed.
- Value of result2(=125) is displayed.

Keys to Dictionary[edit]

Only **IMMUTABLE** objects can be used as keys to dictionaries. Immutable objects are those that cannot be changed. So far, the only mutable objects we've come across are lists and dictionaries.

Trying to use a mutable object as a key causes a **TypeError**.

In short...

Immutable --> Cannot be changed after it is created.

Mutable --> Objects can change their value.

Dictionary --> A built-in Python data type composed of arbitrary keys and values.

Dictionary Functions[edit]

Just like lists, dictionary keys can be assigned to different values. However, unlike lists, a new dictionary can also be assigned a value, not just ones that already exist.

```
>>>squares = {1:1, 2:4, 3:"error", 4:16}

>>>squares[8] = 64

>>>squares[3] = 9

>>>print(squares)
```

Output: {8:64, 1:1, 2:4, 3:9, 4:16}

To determine whether a key is in a dictionary, we can use "**in**" & "**not in**", just like a list.

A useful dictionary method is "**get**". It does the same thing as indexing, but if the key is not found in the dictionary, it returns another specified value instead. ('None', By Default)

>>>pairs = {1:"apple", "orange":[2, 3, 4], True:False}

>>>print(pairs.get("orange")) Will give the output --> **[2, 3, 4]**

>>>print(pairs.get(7)) Will give the output --> **None**

Common Exceptions[edit]

ImportError: An import fails.

IndexError: A list is indexed with an "out-of-range" number.

NameError: An unknown variable is used.

SyntaxError: The code can't be parsed properly.

TypeError: A function is called on a value of an inappropriate type.

ValueError: A function is called on a value of the correct type but with an inappropriate value.

Exception Handling[edit]

To handle exceptions & to call a code when an exception occurs we can use a **try/except** statement.

The try block contains code that might throw an exception. If that exception occurs, the code in the try block stops being executed & the code in except block is run. If no error occurs, the code in the except block doesn't run.

```
>>>try:
        ____________
        ____________

  except ____________:
        ________________
        ________________
```

A try statement can have **different** except blocks to handle different exceptions. Multiple exceptions can also be put into a single except block using parenthesis to have the except block handle all of them.

```
>>>try:
        ____________
        ____________

  except ____________:
        ________________
```

```
    _____________

except (ValueError, TypeError):
    _____________
    _____________
```

An except statement **without** any exception specified will catch all errors. These should be used sparingly, as they can catch unexpected errors & hide programming mistakes. It is particularly useful when dealing with user inputs in a program.

Hence...

Exception handling allows us to gracefully deal with runtime errors.

Can check type of error and take appropriate action based on type.

Can change coding style to exploit exception handling.

When dealing with files and input/output, exception handling becomes very important.

"finally"[edit]

To ensure that some code runs no matter what errors occur, we can use a **finally** statement.
The **finally** statement is placed at the bottom of a try/except statement.

Code within a finally statement always runs after execution of the code in the try, & possibly in the except blocks.

```
>>>try:
    _____________
    _____________

except ___________:
    _____________
    _____________

finally:
    _____________
    _____________
```

Code in a finally statement even runs if an uncaught exception occurs in one of the preceding blocks.

Raising exceptions[edit]

You can raise exceptions by using the **raise** statement.

>>>print(1)

>>>raise ValueError

>>>print(2)

Output:

1

ValueError

You need to specify the type of exception used.

Exceptions can be raised with arguments that give details about them.

For example, **>>>raise NameError("Invalid name !!")**

In except blocks, the raise statement can be used without arguments to **re-raise** whatever exception occurred. raise statement can also be used outside an except block.

>>>try:

```
    num = 5/0
except:
    print("An Error")
raise
```

Output:

An Error

ZeroDivisionError: division by zero

Tuples (Additional Information)[edit]

Tuples are very similar to lists, except that they are immutable(they cannot be changed). also, they are created using parenthesis, rather than square brackets.

>>>words = ("Hello", "world")

We can access the values in tuple with their index. Trying to re-assign a value in a tuple causes a **TypeError**.

Like lists & dictionaries, tuples can also be nested within each other. An empty tuple is created using an empty parenthesis pair.

Tuples can be created without the parenthesis, by just separating the values with commas.

>>>My_tuple = "one", "two", "three"

Tuples are faster than lists, BUT they are Immutable.

Many times, a tuple is used in combination with a dictionary. For example, a tuple might represent a key, because its immutable.

When to use a Dictionary ?[edit]

When we need a "**logical association**" between a key:value pair.

When we need a **fast lookup** for our data, based on a custom key.

When our data is being **constantly modified**. Dictionaries are mutable.

When to use Other Types ?[edit]

Use **lists** if you have a collection of data that does not need random access. Try to choose lists when you need simple, iterable collection that is modified constantly.

Use a **set** if you need uniqueness for the elements.

Use **tuples** when your data cannot change.

Ternary Operator[edit]

Conditional expressions provide the functionality of if statements while using less code. They shouldn't be overused, as they can easily reduce readability, but they are often useful when assigning variables. Conditional expressions are also known as the applications of the ternary operator.

>>>a = 7

>>>**b = 1 if a >= 5 else 42**

>>>print(b)

Output: 1

Assertions[edit]

An assertion is a sanity-check that you can turn on or turn off when you have finished testing the program.

An expression is tested & if the result comes up false, an exception is raised.

Assertions are carried out through the use of **assert** statement.

>>>print(1)

>>>assert 2 + 2 == 4

>>>print(2)

>>>assert 1 + 1 == 3

>>>print(3)

Output:

1

2

AssertionError

Programmers often place assertions at the start of a function to check for valid input, & after a function call to check for valid output.

The assert can take a second argument that is passed to the AssertionError raised if the assertion fails.

>>> temp = -10

>>>assert(temp >= 0), "Colder than 0(Zero)"

Output:

AssertionError: Colder than 0(Zero)

AssertionError exceptions can be caught & handled like any other exception using the try/except statement, but if not handled, this type of exception will terminate the program.

Text Analyzer Program (Involves working with files)[edit]

A function here counts how many times a character occurs in a string.

The other part of the program finds what percentage of the text each character of the alphabet occupies.

```
 1 def count_char(text, char):
 2             count = 0
 3
 4             for c in text:
 5                   if(c == char):
 6                             count += 1
 7              return count
 8
 9 filename = input("Enter a file name: ")
10
11 with open(filename) as f:
12              text = f.read()
13
14 for char in "abcdefghijklmnopqrstuvwxyz":
15           perc = 100 * count_char(text, char)/len(text)
16           print("{0}-{1}%".format(char, round(perc, 2)))
```

Python Enhancement Proposals (PEP)[edit]

Python enhancements proposals (PEP) are suggestions for improvements to the language by experienced Python developers.

PEP 8 is a style guide on the subject of writing readable code. It contains a number of guidelines in reference to variable names, which are summarized here:

--modules should have short, all lower-case names.

--class names should be written in "CapWords" style.

--most variables & function names should be lowercase_with_underscores.

--constants(variables that never change value) should be CAPS_WITH_UNDERSCORES.

--names that would clash with Python keywords(such as "class" or "if") should have a trailing underscore.

PEP 8 also recommends using spaces around operators and after commas to increase readability.

However, whitespace should not be overused. For instance, avoid having any space directly inside any type of brackets.

PEP 8 (Suggestions)[edit]

Other **PEP 8** suggestions include:

--lines shouldn't be longer than 80 characters.

--"from module import *" should be avoided.

--there should only be one statement per line.

It also suggests that you use spaces, rather than tabs, to indent. However, to some extent, this is a matter of personal preference. If you use spaces, only use 4 per line. It's more important to choose one & stick to it.

The most important advice in the PEP is to ignore it when it makes sense to do so. Don't bother with following PEP suggestions when it would cause your code to be less readable, inconsistent with the surrounding code, or not backwards compatible.

However, by & large, following PEP 8 will greatly enhance the quality of your code.

Some other notable PEPs that cover code style are:

PEP 20: The Zen of Python

PEP 257: Style conventions for Docstrings.

Three Doors and a Twist Monte hall[edit]

IN THIS PROGRAM THERE WOULD BE THREE DOORS, THE DOORS COMPRISES OF GOAT IN TWO DOORS AND A CAR IN A SINGLE DOOR. THE PLAYER FIRST SELECTS A DOOR AND THEN THE HOST OPENS A DOOR CONTAINING A GOAT. THE HOST OFFERS A CHOICE TO THE PLAYER THAT HE CAN SWAP HIS OPTION. IT HAS BEEN SEEN THAT THE PLAYER WHO SWAPS HIS/HER CHOICE HAS A HIGHER CHANCE OF WINING THE CAR.

THE ABOVE IS THE OVER VIEW OF THE GAME. LETS SEE THE PROGRAM THROUGH WHICH WE CAN PLAY THE GAME OF **MONTE HALL**

import random (this is used because we use random functions in the program)

n= int(input("how many doors u want ??"))

door = [0]*n (the door is multiplied with n as there are n doors)

goatdoor = [0]*n-1 (since a door contains car the other doors must contain goat)

swap = 0 (we initialize the swap as zero)

dont_swap = 0 (we initialize don't swap as zero)

j = 0

while(j<20): (It gives the number of trials that we can play)

x = random.choice(0,n) (there are three doors so we initialize it)

door [t] = "Audi" for x in range (0,n+1):

```
   if (x==t):
     continue
   else:
     door[x] = goat
     goatdoor.append(x)
```

choice = int(input("please enter any door number to open))

door_open = random.choice(goatdoor). # it opens a door with goat randomly

while(door_open=choice). # these two values must not be equal

door_open = random.choice(goatdoor)

1. To swap

sw = input(" Do u want to swap? Y/N")

if (sw = Y): # in this case player is willing to swap his choice

```
 if(door[choice] = goat)  ( # the player chooses a goat door first and
he swaps
```

```
    print(" bumper prizer!!!")                    -  if he wins the
game )
    swap = swap+1
else:

   print(" sorry better luck next time")
```

1. Don't swap

if(door(choice)= goat): (the player has chosen a goat door and refused to swap)

```
  print("better luck next time")
```

else:

```
  print(" bumper prize!!!")
  dontswap = dontswap+1   (the don't swap is incremented )
  j = j+1
```

print(swap) print (dontswap)

1. The player who swaps has a higher chance of winning

Please do visit the Wikipedia link for more interesting information https://en.m.wikipedia.org/wiki/Monty_Hall_problem

Lecture Notes:Week 6

From JOCWiki

Contents

[hide]

- 1 **Cryptography**
  - 1.1 Characteristics of Modern Cryptography
  - 1.2 Terminologies of Cryptography
  - 1.3 **Substitution Cipher**
  - 1.4 Code with Explanation
  - 1.5 Program for Substitution Cipher
  - 1.6 String Formatting
  - 1.7 String Processing
  - 1.8 Strip Whitespace
  - 1.9 Searching for Text
  - 1.10 Joining Strings
  - 1.11 Converting Case
  - 1.12 Double Strength Encryption
  - 1.13 Cryptography Packages
  - 1.14 **Algorithm for Reverse Cipher**
  - 1.15 **Algorithm for Caesar Cipher**
  - 1.16 **Hybrid Cryptography**
  - 1.17 **Explanation of ROT13 algorithm**
- 2 **Tic Tac Toe**
  - 2.1 NumPy

- 2.2 Tic Tac Toe **Code as per the video lecture:**
- 2.3 Tic Tac Toe **Code :**
- 2.4 **Can we use dictionary for *"Tic Tac Toe"*?**

- 3 **Recursion**
  - 3.1 Binary Search Code: with small correction
  - 3.2 Fibonacci Sequence Code
  - 3.3 4 Different Ways of Fibonacci in Python
  - 3.4 Fibonacci Sequence using Memoization as a Decorator
- 4 **Functional Programming**
  - 4.1 Generators
  - 4.2 Lambdas
  - 4.3 Filter
  - 4.4 Decorators

Cryptography[edit]

Cryptography is the practice and study of techniques for secure communication in the presence of third parties called adversaries.

Ciphers: In cryptography, a cipher is an algorithm(or technique) used for performing encryption or decryption to ensure secure data communication.It is a method used to hide words(or text) by replacing original letters with other letters or symbols.

Encryption: Encryption is the process of converting information or data into a secret(or unreadable) code to prevent unauthorized users.

Decryption: Decryption is the reverse process of encryption.It is used to convert(or decode) the encrypted data into readable form.

Cryptography is the art of communication between two users via coded messages. The science of cryptography emerged with the basic motive of providing security to the confidential messages transferred from one party to another.

Characteristics of Modern Cryptography[edit]

The basic characteristics of modern cryptography are as follows −

-- It operates on bit sequences.

-- It uses mathematical algorithms for securing the information.

-- It requires parties interested in secure communication channel to achieve privacy.

Terminologies of Cryptography[edit]

The frequently used terms in cryptography are as follows:

Plain Text

The plain text message is the text which is readable and can be understood by all users. The plain text is the message which undergoes cryptography.

Cipher Text

Cipher text is the message obtained after applying cryptography on plain text.

Encryption

The process of converting plain text to cipher text is called encryption. It is also called as encoding.

Decryption

The process of converting cipher text to plain text is called decryption. It is also termed as decoding.

Substitution Cipher[edit]

It involves replacement of letters of the plaintext message by other letters or symbols.

Caesar Cipher: It is the earliest known substitution cipher used for sending secret messages by Julius Caesar. It replaces or shifts each letter in the message by certain number of places, say by the next 3rd letter.

For example,

```
Message: MEET ME
Cipher Text: PHHW PH
```

Code with Explanation[edit]

Posted By: Arunkumar | Doubts on this topic?

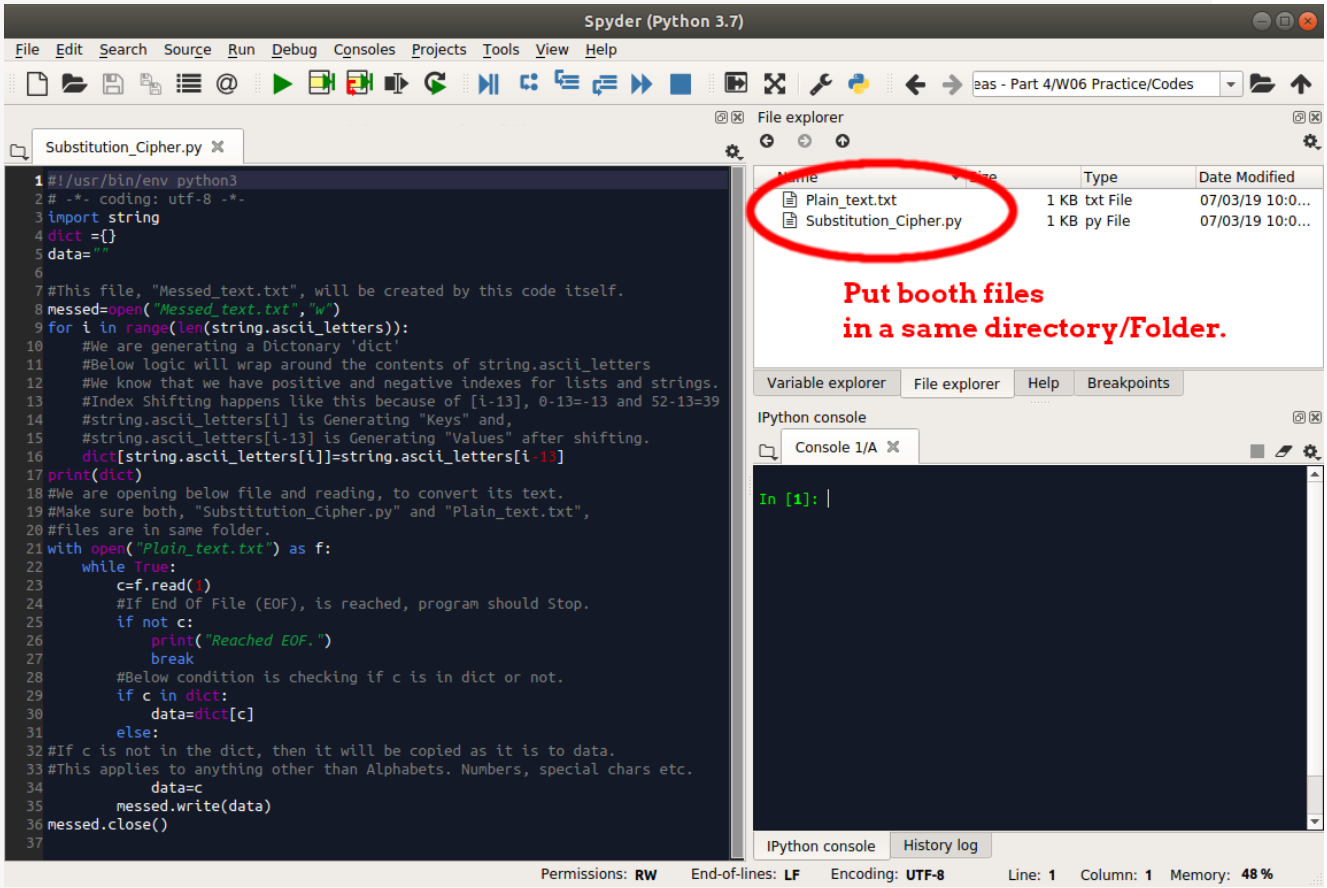
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import string
dict ={}
data=""

#This file, "Messed_text.txt", will be created by this code itself.
messed=open("Messed_text.txt","w")

for i in range(len(string.ascii_letters)):
    """
    We are generating a Dictonary 'dict'
    Below logic will wrap around the contents of
string.ascii_letters
```

```
    We know that we have positive and negative indexes for lists and strings.
    Index Shifting happens like this because of [i-13], 0-13=-13 and 52-13=39
    string.ascii_letters[i] is Generating "Keys" and,
    string.ascii_letters[i-13] is Generating "Values" after shifting.
    """
    dict[string.ascii_letters[i]]=string.ascii_letters[i-13]

#If you want to see the dictionary used for conversion, uncomment below print comment.
#print(dict)
"""
We are opening below file and reading, to convert its text.
Make sure both, "Substitution_Cipher.py" and "Plain_text.txt",
files are in same folder.
"""
with open("Plain_text.txt") as f:
    while True:
        c=f.read(1)
        #If End Of File (EOF), is reached, program should Stop.
        if not c:
            print("Reached EOF.")
            break
        #Below condition is checking if c is in dict or not.
        if c in dict:
            data=dict[c]
        else:
        """
        If c is not in the dict, then it will be copied as-it-is to
        data. This applies to anything other than Alphabets.
        For example... Numbers, special chars etc.
        """
            data=c
        messed.write(data)
```

```
46 messed.close()
```

"Plain_text.txt" Content:

> Once upon a time, in a jungle there lived a jackal by the name of Gomaya. One day, he was very hungry and was wandering in search of food. While wandering, he came across a battle field. There he saw a big drum lying under a tree. When the wind blew, a tender branch grown at the root of the tree struck the drum producing sound of a drum beat. The jackal examined the drum from all sides and then beat the drum with his front paws. The drum made a sound. Now the jackal thought that there might be some other small animal inside the drum and that would make a very tasty meal for him. But he found the top of the drum too tough to tear off.




```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import string
dict ={}
data=""

#This file, "Messed_text.txt", will be created by this code itself.
messed=open("Messed_text.txt","w")
for i in range(len(string.ascii_letters)):
    #We are generating a Dictonary 'dict'
    #Below logic will wrap around the contents of string.ascii_letters
    #We know that we have positive and negative indexes for lists and strings.
    #Index Shifting happens like this because of [i-13], 0-13=-13 and 52-13=39
    #string.ascii_letters[i] is Generating "Keys" and,
    #string.ascii_letters[i-13] is Generating "Values" after shifting.
    dict[string.ascii_letters[i]]=string.ascii_letters[i-13]
print(dict)
#We are opening below file and reading, to convert its text.
#Make sure both, "Substitution_Cipher.py" and "Plain_text.txt",
#files are in same folder.
with open("Plain_text.txt") as f:
    while True:
        c=f.read(1)
        #If End Of File (EOF), is reached, program should Stop.
        if not c:
            print("Reached EOF.")
            break
        #Below condition is checking if c is in dict or not.
        if c in dict:
            data=dict[c]
        else:
#If c is not in the dict, then it will be copied as it is to data.
#This applies to anything other than Alphabets. Numbers, special chars etc.
            data=c
        messed.write(data)
messed.close()
```

Program for Substitution Cipher[edit]

- String is used to represent text in python.String comprises of spaces, numbers, characters and alphabets.
- To use string in a program we have to import a library called string.
  syntax: import string

- Note that a string is nothing but a list of characters which may include space and special characters.

- A string should always be initialized within double quotes.
- List indexing always begins from 0.
- import string
  print(string.ascii_letters)

- The concatenation of the ascii_lowercase and ascii_uppercase constants.
  Prints abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

```
word = "Joy of Computing"  #string can be enclosed with double or single quotes
print(word)
```

```
Joy of Computing
```

```
# String inbuilt functions
print(word.lower())
print(word.upper())
print(len(word))

# Slicing the string
print(word[0:5]) #start index to end index
```

```
joy of computing
JOY OF COMPUTING
16
Joy o
```

```
# Coverting string to a list of characters
print(list(word))
```

```
['J', 'o', 'y', ' ', 'o', 'f', ' ', 'C', 'o', 'm', 'p', 'u', 't', 'i', 'n', 'g']
```

String Formatting[edit]

String formatting provides a more powerful way to embed non-strings within strings. String formatting uses a string's **format** method to substitute a number of arguments in the string.

>>>nums = [4, 5, 6]

>>>msg = "Numbers: {0} {1} {2}".format(nums[0], nums[1], nums[2])

>>>print(msg)

Output: **Numbers: 4 5 6**

- Each argument of the format function is placed in the string at the corresponding position, which is determined by using the curly braces {}.
- String formatting can also be done with named arguments.

String Processing[edit]

Easy to read & write text files.

String processing functions make it easy to analyse & transform contents.

Strip Whitespace[edit]

s.rstrip() removes trailing whitespaces.

>>>for line in contents:

```
    s = line.rstrip()
```

s.lstrip() removes leading whitespaces.

s.strip() removes leading and trailing whitespaces.

Searching for Text[edit]

s.find(pattern) returns first position in **s** where **pattern** occurs, **-1** if no occurrence of pattern.

s.find(patten, start, end)

Search for **pattern** in slice **s[start: end]**

s.index(pattern)

s.index(pattern, l, r)

Joining Strings[edit]

Recombine a list of strings using a separator.

columns = s.split(",")

joinstring = ","

csvline = joinstring.join(columns)

date = "16"

month = "08"

year = "2021"

today = "-".join([date, month, year])

Converting Case[edit]

s.captalize() -- returns a new string with first letter Uppercase, rest lower.

s.lower() -- Converts all uppercase letters to lowercase.

s.uppercase() -- Converts all lowercase letters to uppercase.

s.swapcase() -- The swapcase() method returns a copy of the string in which all the case-based characters have had their case swapped.

Double Strength Encryption[edit]

Double strength encryption, also called as multiple encryption, is the process of encrypting an already encrypted text one or more times, either with the same or different pattern/algorithm.

The other names for double strength encryption include cascade encryption or cascade ciphering.

Cryptography Packages[edit]

Python includes a package called cryptography which provides cryptographic recipes and primitives. There are various packages with both high level recipes & low level interfaces to common cryptographic algorithms such as **symmetric ciphers**, **message digests** & **key derivation function**.

Algorithm for Reverse Cipher[edit]

The algorithm for reverse cipher holds the following features --

- Reverse cipher uses a pattern of reversing the string of plain text to convert as cipher text.
- The process of encryption and decryption is same.
- To decrypt cipher text, the user simply needs to reverse the cipher text to get the plain text.

Drawback

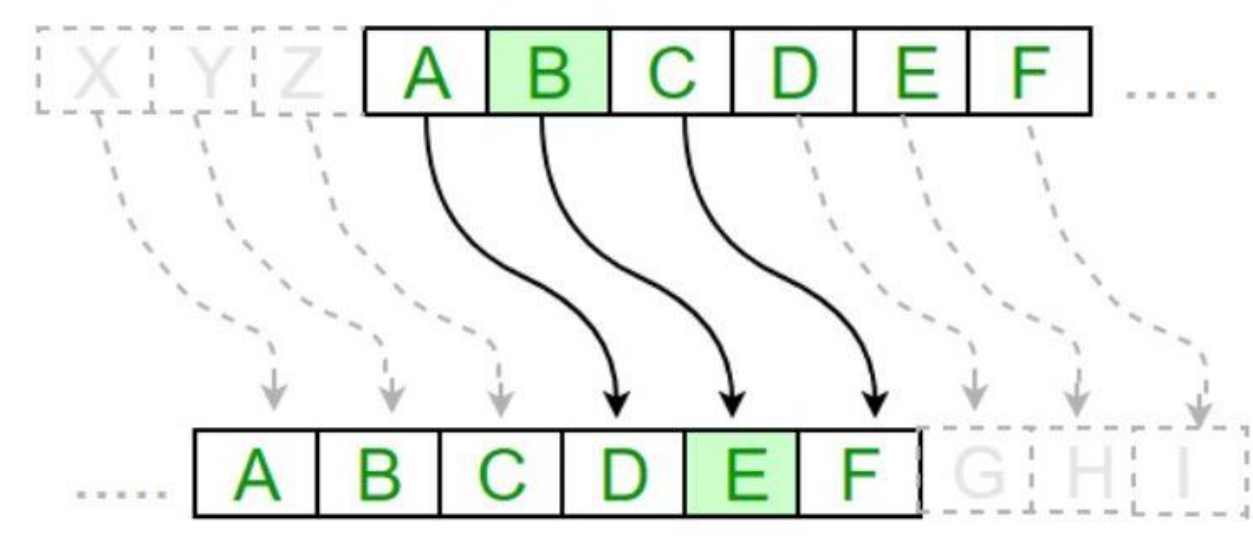
The major drawback of reverse cipher is that it is very weak. A hacker can easily break the cipher text to get the original message. Hence, reverse cipher is not considered as good option to maintain secure communication channel.

Algorithm for Caesar Cipher[edit]

The algorithm for Caesar Cipher holds the following features --

- Caesar Cipher technique is simple & easy method of encryption technique.
- It is simple type of substitution cipher.
- Each letter of plain text is replaced by a letter with some fixed number of positions down with alphabet.

The following diagram depicts the working of Caesar cipher algorithm implementation −



Hacking of Caesar Cipher Algorithm

The cipher text can be hacked with various possibilities. One of such possibility is Brute Force Technique, which involves trying every possible decryption key. This technique does not demand much effort and is relatively simple for a hacker.

Hybrid Cryptography[edit]

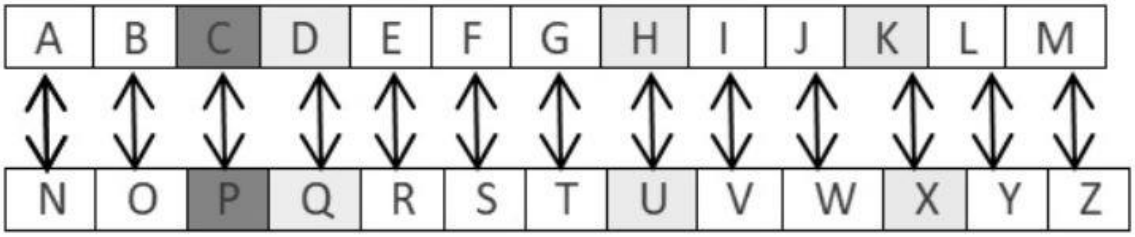
Hybrid cryptography is the process of using multiple ciphers of different types together by including benefits of each of the cipher. There is one common approach which is usually followed to generate a random secret key for a symmetric cipher and then encrypt this key via asymmetric key cryptography.

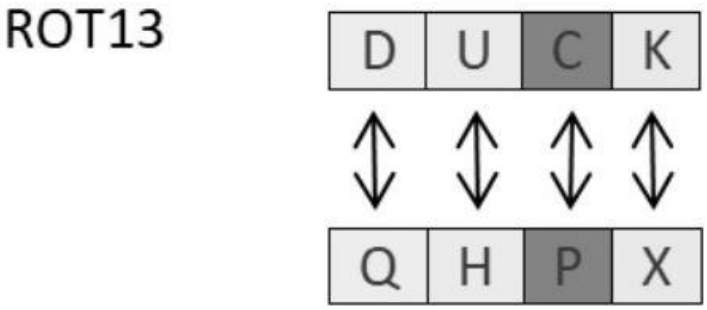
Due to this pattern, the original message itself is encrypted using the symmetric cipher and then using secret key. The receiver after receiving the message decrypts the message using secret key first, using his/her own private key and then uses the specified key to decrypt the message.

Explanation of ROT13 algorithm[edit]

ROT13 cipher refers to the abbreviated form **Rotate by 13 places**. It is a special case of Caesar Cipher in which shift is always 13. Every letter is shifted by 13 places to encrypt or decrypt the message.

Pictorial Representation





Program code for ROT13 algorithm:

```
from string import maketrans


rot13trans =
maketrans('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz',
   'NOPQRSTUVWXYZABCDEFGHIJKLMnopqrstuvwxyzabcdefghijklm')


# Function to translate plain text
def rot13(text):
```

```
    return text.translate(rot13trans)
def main():
    txt = "ROT13 Algorithm"
    print rot13(txt)

if __name__ == "__main__":
    main()
```

Drawback

The ROT13 algorithm uses 13 shifts. Therefore, it is very easy to shift the characters in the reverse manner to decrypt the cipher text.

Analysis of ROT13 Algorithm

ROT13 cipher algorithm is considered as special case of Caesar Cipher. It is not a very secure algorithm and can be broken easily with frequency analysis or by just trying possible 25 keys whereas ROT13 can be broken by shifting 13 places. Therefore, it does not include any practical use.

Tic Tac Toe[edit]

Through this program we will see how arrays can be used.

NumPy[edit]

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, & tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including:

-- A powerful N-dimensional array object.

--Sophisticated (broadcasting) functions.

--Tools for integrating C/C++ & Fortran code.

--Useful linear algebra, Fourier transform, & random number capabilities.

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.

Arbitrary data-types can be defined using NumPy which allows NumPy to seamlessly & speedily integrate with a wide variety of databases.

Tic Tac Toe Code as per the video lecture:[edit]

```
1 import numpy
2 board = numpy.array([['-','-','-'],['-','-','-'],['-','-','-']])
3 p1s='X'
4 p2s='O'
5
```

```
 6 def check_rows(symbol):
 7     for r in range(3):
 8         count=0
 9         for c in range(3):
10             if board[r][c]==symbol:
11                 count=count+1
12         if count==3:
13             print(symbol,"won")
14             return True
15     return False
16 def check_cols(symbol):
17     for c in range(3):
18         count=0
19         for r in range(3):
20             if board[r][c]==symbol:
21                 count=count+1
22         if count==3:
23             print(symbol,"won")
24             return True
25     return False
26
27 def check_diagonals(symbol):
28     if board[0][2] == board[1][1] and board[1][1] == board[2][2] and
board[1][1]==symbol:
29         print(symbol, 'won')
30     if  board[0][0] == board[1][1] and board[1][1] == board[2][2]
and board[1][1]==symbol:
31          print(symbol, 'won')
32          return True
33     return False
34
35 def won(symbol):
36     return check_rows(symbol) or check_cols(symbol) or
check_diagonals(symbol)
37
38 def place(symbol):
```

```
39      print(numpy.matrix(board))
40      while(1):
41          row = int(input("Enter row - 1 2 3:"))
42          column = int(input("Enter column - 1 2 3:"))
43          if row>0 and row<4 and column>0 and column<4 and board[row-
1][column-1]=='-':
44              break
45          else:
46              print('Invalid input, Please enter again')
47
48      board[row-1][column-1]=symbol
49
50  def play():
51      for turn in range(9):
52          if turn%2==0:
53              print('X turn')
54              place(p1s)
55              if won(p1s):
56                  break
57          else:
58              print('O turn')
59              place(p2s)
60              if won(p2s):
61                  break
62
63      if not(won(p1s)) and not(won(p2s)):
64          print("Draw")
65
66  play()
```

Tic Tac Toe Code :[edit]

Posted By: Arunkumar | Doubts on this topic?

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
```

```
3 #We Use arrays for this game, for that we need to import numpy
4 import numpy
5 Nboard=numpy.array([['11','12','13'],
6                     ['21','22','23'],
7                     ['31','32','33']])
8
9 Gboard=numpy.array([['_','_','_'],
10                    ['_','_','_'],
11                    ['_','_','_']])
12 playerSymbol1='X'
13 playerSymbol2='O'
14
15 def check_rows(symbol):
16     for r in range(3):
17         repeated=0
18         for c in range(3):
19             if Gboard[r][c]==symbol:
20                 repeated += 1
21         if repeated==3:
22             return True
23     return False
24
25 def check_cols(symbol):
26     for c in range(3):
27         repeated=0
28         for r in range(3):
29             if Gboard[r][c]==symbol:
30                 repeated += 1
31         if repeated==3:
32             return True
33     return False
34
35 def check_diag(a,b,symbol):
36     repeated=0
37     for r,c in zip(a,b):
38         if Gboard[r][c]==symbol:
```

```
39              repeated += 1
40          if repeated==3:
41              return True
42      return False
43
44  def won(sym):
45      a=[0,1,2]
46      return (check_rows(sym) or check_cols(sym) or
47              check_diag(a,a,sym) or
check_diag(a,list(reversed(a)),sym))
48
49  def place(symbol):
50      print('{0: ^17}'.format("Place numbers"))
51      print(numpy.matrix(Nboard),"\n==================\n")
52      print(numpy.matrix(Gboard))
53      while(1):
54          row=int(input('Enter Row - 1 or 2 or 3: '))
55          col=int(input('Enter Column - 1 or 2 or 3: '))
56          if row>0 and row<4 and col>0 and col<4 and Gboard[row-
1][col-1]=='_':
57              break
58          else:
59              print('Invalid input. Please enter again.')
60      Gboard[row-1][col-1]=symbol
61
62  def play():
63      for turn in range(9):
64          if turn%2==0:
65              print("\n",'{0: ^17}'.format("X\'s turn"))
66              place(playerSymbol1)
67              if won(playerSymbol1):
68                  print(playerSymbol1,' Won!')
69                  break
70          else:
71              print("\n",'{0: ^17}'.format("O\'s turn"))
72              place(playerSymbol2)
```

```
73              if won(playerSymbol2):
74                  print(playerSymbol2,' Won!')
75                  break
76      if not(won(playerSymbol1)) and not(won(playerSymbol2)):
77          print('This game is a Draw!')
78
79  play()
```

Can we use dictionary for *"Tic Tac Toe"*?[edit]

Let's explore/experiment...
We are, kind of, over using the `won()` function. Can't we use it once for every change and store another variable? For that we need two `won` variables for both players. This is where, using Dictionary, makes sense. Lets replace few variables in the above program.

- Instead of *`playerSymbol1`* and *`playerSymbol2`* (line 12, 13), lets use **`player1={'symbol':'X','won':False}`** and **`player2={'symbol':'X','won':False}`**

```
12 player1={'symbol':'X','won':False}
13 player2={'symbol':'X','won':False}
```

- `print('{0: ^17}'.format("O\'s turn"))` is centre aligning "**O's turn**" in 17 characters space.
- We need to change in `play()` too. Observe `player1['won']=won(player1['symbol'])`, we are calling `won()` function and saving the state into a *dictionary variable*. We will do as below.

```
62  def play():
63      for turn in range(9):
64          if turn%2==0:
65              print("\n",'{0: ^17}'.format("X\'s turn"))
66              place(player1['symbol'])
67              player1['won']=won(player1['symbol'])
68              if player1['won']:
69                  print(player1['symbol'],' Won!')
70                  break
71          else:
72              print("\n",'{0: ^17}'.format("O\'s turn"))
73              place(player2['symbol'])
74              player2['won']=won(player2['symbol'])
75              if player2['won']:
```

```
76                  print(player2['symbol'],' Won!')
77                  break
78      if not(player1['won']) and not(player2['won']):
79          print('This game is a Draw!')
```

Recursion[edit]

Please visit here: http://sccilabs.org/jocwiki/index.php/Lecture_Notes:Week_4#Recursion

Binary Search Code: with small correction[edit]

Recursion Part 05 video.
There was a problem in the code from the video. One of the checks is missing. When search value is less than all the numbers in the list, program gets stuck with mid=0. Hence it is failing with that error, **"RecursionError: maximum recursion depth exceeded in comparison"**. So, we just need to add another condition in **elif** part (line 15) in the function, `indexMid>0`.

```
 1 #!/usr/bin/env python3
 2 # -*- coding: utf-8 -*-
 3 def binary_search(searchList, searchFor, indexStart, indexEnd):
 4     # Base Case : if 1 element is left in the list.
 5     if indexStart==indexEnd:
 6         if searchList[indexStart]==searchFor:
 7             return indexStart
 8         else:
 9             return -1
10     else:
11         #we will divide the list into half and search
12         indexMid=int((indexStart+indexEnd)/2)
13         if searchList[indexMid]==searchFor:
14             return indexMid
15         elif searchList[indexMid]>searchFor and indexMid>0:
16             #Left Half
17             return binary_search(searchList, searchFor, indexStart,
indexMid-1)
18         else:
19             #Right Half
```

```
20             return binary_search(searchList, searchFor, indexMid+1,
indexEnd)
21 """
22 For this program, I intentionally used different variable names for
functions
23 (above) and the program (below). So that students can understand
that we can
24 use different variable names too.
25 """
26 n=[20,60,81,34,76]
27 n.sort()
28 s=int(input('Enter Search number: '))
29 result=binary_search(n, s, 0, len(n)-1)
30 if result==-1:
31     print(s, 'not found!')
32 else:
33     print(s, 'was found at position', result+1)
```

Fibonacci Sequence Code[edit]

```
def Fibonacci(n):
    if(n <= 1):
        return n
    else:
        return(fibonacci(n-1) + fibonacci(n-2))

n = int(input("Enter number of terms:"))
print("Fibonacci sequence:")
for i in range(n):
    print fibonacci(i)
```

4 Different Ways of Fibonacci in Python[edit]

1. Using looping technique

```
def fib(n):
       a,b = 1,1
       for i in range(n-1):
```

```
        a,b = b,a+b
        return a

print fib(5)
```

2. Using Recursion

```
def Fibonacci(n):
    if(n <= 1):
        return n
    else:
        return(fibonacci(n-1) + fibonacci(n-2))

n = int(input("Enter number of terms:"))

print("Fibonacci sequence:")
for i in range(n):
    print fibonacci(i)
```

3. Using generators

```
a,b = 0,1
def fibI():
      global a,b
      while True:
      a,b = b, a+b
      yield a
      f=fibI()
      f.next()
      f.next()
      f.next()
      f.next()

print f.next()
```

4. Using memoization

```
def memoize(fn, arg):
    memo = {}
    if arg not in memo:
```

```
    memo[arg] = fn(arg)
    return memo[arg]

def fib(n):
       a,b = 1,1
       for i in range(n-1):
       a,b = b,a+b
       return a

fibm = memoize(fib,5)

print(fibm)
```

Fibonacci Sequence using Memoization as a Decorator[edit]

```
class Memoize:
      def __init__(self, fn):
                    self.fn = fn
                    self.memo = {}

       def __call__(self, arg):
             if arg not in self.memo:
                    self.memo[arg] = self.fn(arg)
       return self.memo[arg]

@Memoize
def fib(n):
    a,b = 1,1
    for i in range(n-1):
          a,b = b,a+b
          return a

print fib(5)
```

Functional Programming[edit]

A function is a block of organised, reusable code that is used to perform a single, related action. Functions provide better modularity for your program & a high degree of code reusing.

Functional programming is a style of programming that is based around functions. A key part of functional programming is higher order functions.

Pure Functions:

Pure functions have no side effects & return a value that depends on their arguments.

1. Pure_function

```
>>>def pure(x, y):

        temp = x + 2 * y
        return temp / (2 * x + y)
```

Pure functions are:

-- Easier to reason about & test.

-- More efficient. Once the function has been evaluated for an input, the result can be stored & referred to the next time the function of that inout is needed, reducing the number of times the function is called. This is called **memoization**

{In computing, **memoization** is an optimization technique used primarily to speed up computer programs by storing the results of expensive function calls and returning the cached result when the same inputs occur again. }

-- Easier to run in parallel.

The main advantage of using only pure functions is that they majorly complicate the otherwise simple task of input/output (I/O), since this appears to inherently require side effects. They can also be more difficult to write in some functions.

Generators[edit]

Generators are a type of iterable, like lists or tuples.

Unlike lists, they don't allow indexing with arbitrary indices, but they can still be iterated with for loops.

They can be created using functions & the "**yield**" statement.

```
>>>def count():

    i = 5
        while(i>0):
              yield i
              i-=1

>>>for i in count():

          print(i)
```

Output

5

4

3

2

1

The **yield** statement is used to define a generator, replacing the return of a function to provide a result to its caller WITHOUT destroying local variables.

Iterable -- a container object capable of returning its members, one at a time.

yield -- returns a value from a generator function.

generator -- a function which returns an iterator.

Due to the fact that they yield one item at a time, generators don't have the memory restrictions of the lists. In fact, they can be infinite. In short, generators allow us to declare a function that behaves like an iterator, i.e., it can be used in a for loop.

Finite generators can be converted into lists by passing them as arguments to the list function.

Using generators results in improved performance, which is the result of the lazy (on demand) generation of values, which translates to lower memory usage. Furthermore, we do not need to wait until all elements have been generated before we start to use them.

Lambdas[edit]

Creating a function normally (using def) assigns it to a variable automatically. This is different from the creation of other objects such as strings & integers -- which can be created on the fly, without assigning them to a variable.

The same is possible with functions, provided that they are created using **lambda** syntax. Functions created this way are known as **anonymous**.

This approach is most commonly used when passing a simple function as an argument to another function.

```
>>>def my_func(f, arg):


          return f(arg)
>>>my_func(lambda x:2 * x * x, 5)
```

Lambda -- a short hand to create anonymous functions.

Lambda functions get their name from lambda calculus, which is a model of computation invented by **Alonzo Church**.

Lambda functions aren't as powerful as named functions. They can only do things that require a single expression -- usually equivalent to a single line of code.

>>>print((lambda x: x2 + 5*x +4)(-4))**

Lambda functions can be assigned to variables & used like normal functions.

Filter[edit]

The function **filter** filters an iterable by removing items that don't match a predicate (a function that returns a Boolean).

>>>nums = [11, 22, 33, 44, 55]

```
res = list(filter(lambda x: x % 2 == 0, nums))
print(res)
```

Output: [22, 44]

Like map, the result has to be explicitly converted to a list if we want to print it.

Decorators[edit]

Decorators provide a way to modify functions using other functions. This is ideal when we need to extend the functionality of functions that we don't want to modify.

Decorator -- a function that modifies another function or method.

Parameter -- a named entity in a function (or method) definition that specifies an argument that the function can accept.

We decorate our function by replacing the variable containing the function with a wrapped version.

>>>def print_text():

```
    print("Hello world!")
```

print_text = décor(print_text)

This pattern can be used at any time, to wrap any function. Python provides support to wrap a function in a decorator by pre-pending the function definition with a decorator name & the **@** symbol.

>>>@decor

>>>def print_text():

```
    print("Hello!")
```

A single function can have multiple decorators.

Lecture Notes:Week 7

From JOCWiki

Contents

[hide]

- 1 SNAKES AND LADDER GAME
- 2 GUESS THE MOVIE (Revision Week 4)
- 3 Turtle Module
- 4 Plotting Using Turtle
- 5 Basic Shapes Using Turtle Module
- 6 Data Structure: SETS
- 7 What is a CSV file ?
- 8 CSV files
- 9 Working with the CSV Module
- 10 CSV Functions
- 11 GMPLOT
- 12 Geocoding
- 13 IterTools
- 14 Turtle Program to Print "Rainbow Benzene"
- 15 Python Imaging Library (PIL)
- 16 Operations With Images
- 17 Animated Spiral with Colors added
- 18 An Amazing Design With Turtle

SNAKES AND LADDER GAME[edit]

Snake and Ladder is a game which involves a whole lot of randomness and there is no place for logic and thinking. The game is played between two or more players on a game-board having numbered from 1 to 100, gridded squares. A number of "ladders" and "snakes" are pictured on the board, each connecting two specific board squares.

It is possible to trace the origins of the game back to the 2nd Century B.C. as the Indian game of '***Paramapada Sopanam***'— "The Ladder to Salvation."

It is originated in India and is popular all over the World.

1. Each player has a pawn of different color placed on number 1 on game-board.
2. Each player roll a die one by one and move pawn forward accordingly(no. of steps forward = no. on die).
3. If a player encounter 6 on die ,will move pawn 6 steps forward along with getting another chance to roll a die.
4. If player's pawn land on a ladder will climb up to the height of the ladder and if pawn land on snake's mouth will move downwards till the tail of the snake.
5. One who reaches 100 first will be a Winner.

Snakes and Ladders was brought to the UK in 1892 and was commercially published for the first time in the USA by Milton Bradley in 1943 as "Chutes and Ladders".

.

GUESS THE MOVIE (Revision Week 4)[edit]

```
import random
movies = ['tarzan','jumangi','titanic','matrix','robot','simbaa','kedarnath','andhadhun']
film  = list(random.choice(movies))
guess = []
for i in range(len(film)):
    guess.append('_')
for i in guess:
    print(i,end=" ")
while '_' in guess :
    alpha = input("Guess an alphabet")
    while alpha in film:
        index = film.index(alpha)
        film[index]=' '
        guess[index]=alpha
    for i in guess:
        print(i,end=" ")

print("\nYou have successfully guessed the film")
```

Turtle Module[edit]

"Turtle" is a Python feature like a drawing board, which lets us command a turtle to draw all over it! We can use different functions which can move the turtle around.

Commonly used turtle methods are:

- **Turtle()** -- creates and returns a new turtle object.
- **forward(x)** -- moves the turtle forward by the specified amount.
- **backward(y)** -- moves the turtle backward by the specifies amount.
- **right(angle)** -- turns the turtle clockwise.
- **left(angle)** -- turns the turtle anti-clockwise.
- **penup()** -- picks up the turtle's pen.
- **pendown()** -- puts down the turtle's pen.
- **color(name)** -- changes the colour of the turtle's pen.
- **fillcolor(name)** -- used to fill a colour in a polygon.
- **heading()** -- returns the current heading.
- **position()** -- returns the current position.
- **goto(x, y)** -- moves the turtle to position (x, y).
- **begin_fill()** -- remember the starting point for a filled polygon.

- **end_fill()** -- close the polygon and fill with the current colour.
- **dot()** -- leave the dot at the current position.
- **stamp()** -- leaves an impression of a turtle shape at the current location.

Plotting Using Turtle[edit]

To make use of the turtle methods and functionalities, we need to import turtle. Turtle comes packed with the standard Python package and need not be installed externally.

The roadmap for executing a turtle program follows 4 steps:

- 1. Import the turtle module.
- 2. Create a turtle to control.
- 3. Draw around using the turtle methods.
- 4. Run turtle.done()

Basic Shapes Using Turtle Module[edit]

Shape: SQUARE

import turtle t = turtle.Turtle()

for i in range(4):

```
    t.forward(50)
    t.right(90)
```

turtle.done()

Shape: STAR

import turtle star = turtle.Turtle()

for i in range(50):

```
    star.forward(50)
    star.right(144)
```

turtle.done()

Shape: HEXAGON

import turtle polygon = turtle.Turtle()

num_sides = 6 side = 70 angle = 360.0/num_sides

for i in range(num_sides):

```
    polygon.forward(side)
    polygon.right(angle)
```

```
turtle.done()
```

Data Structure: SETS[edit]

Sets are data structures, similar to lists or dictionaries. They are created using curly braces, or the set function. They share some functionality with lists, such as use of "in" to check whether they contain a particular item.

>>>num_set = {1, 2, 3, 4, 5}

>>>word_set = set(["hello", "world"])

>>>print(3 in num_set) --> **True**

>>>print("hello" not in word_set) --> **False**

To create an empty set, we must use **set()** as **{}**creates an empty dictionary.

Sets differ from lists in several ways, but share several list operations.

They are unordered, which means that they can't be indexed. **They cannot contain duplicate elements.**

Due to the way they are stored, its faster to check whether an item is a part of a set, rather than a art of a list.

Instead of using "append()" to add to a set, use **add()**.

The method **remove()** removes a specific element from a set; **pop()** removes an arbitrary element.

The basic use of sets include membership testing & the elimination of duplicate entries.

Sets can be combined using mathematical operations.

The **union** operator | combines two sets to form a new one containing items in either.

The **intersection** operator **&** gets items ONLY in BOTH.

The **difference** operator - gets items in the first set but not in the second.

The **symmetric difference** operator ^ gets items in EITHER set, BUT NOT both.

What is a CSV file ?[edit]

CSV files are used to store a large number of variables - or data. They are incredibly simplified spreadsheets - think excel - only the content is stored in plain text.

And the CSV module is a built-in function that allows Python to parse these types of files.

The text inside a CSV file is laid out in rows, and each of those has columns, all separated by commas. Every line in the file is a row in the spreadsheet, while the commas are used to define and separate cells.

CSV files[edit]

A csv file is a text file that stores tabular data separated by comma ',' where csv stands for comma separated values. csv files have .csv extension.

Reading a csv file:

import csv

with open('data.csv') as file:

```
      reader = csv.reader(file) #reader() method returns an object
```

which helps to iterate rows in a csv file
for row in reader: print(row)

file.close()

Output: ['roll_no', 'name']

['1', 'abc']

['2', 'pqr']

['3', 'xyz']

Working with the CSV Module[edit]

To pull information from CSV files you use loop and split methods to get the data from individual columns.

The CSV module explicitly exists to handle this task, making it much easier to deal with CSV formatted files.

This becomes especially important when you are working with data that's been exported from actual spreadsheets and databases to text files. This information can be tough to read on its own.

CSV Functions[edit]

The CSV module includes all the necessary functions built in. They are:

- **csv.reader**
- **csv.writer**
- **csv.register_dialect**
- **csv.unregister_dialect**
- **csv.get_dialect**
- **csv.list_dialects**
- **csv.field_size_limit**

GMPLOT[edit]

Command to install gmplot: pip install gmplot

Simple map with a line connecting cities:

```
from gmplot import gmplot


# latitudes and longitudes of three cities
lats = [ 31.326000,31.224000,30.9010]

longs = [ 75.576200,75.770800,75.8573]


#shows the center of map
gmap = gmplot.GoogleMapPlotter(31.268900,75.701500, zoom=12)


# scatter points on the google map
gmap.scatter( lats, longs, size=40 )


# draws a line between the scattered points

# here mediumblue is the color of the line and edge_width is thickness
gmap.plot(lats, longs,'mediumblue', edge_width = 4)

gmap.draw( "map.html" )
```

Note: We can also provide path of the html file as following:

```
gmap.draw( "C:\\Users\\usr\\.spyder-py3\\map.html" )
```

Use double backslash because single \ acts as escape character.

Geocoding[edit]

gmplot contains a simple wrapper around Google's geocoding service enabling map initilization to the location of your choice. Rather than providing latitude, longitude, and zoom level during initialization, grab your gmplot instance with a location:

gmap = gmplot.GoogleMapPlotter.from_geocode("India")

Plot Types

- Polygons with fills - **plot**
- Drop pins - **marker**
- Scatter points - **scatter**
- Grid lines - **grid**
- Heatmaps - **heatmap**

IterTools[edit]

The module **itertools** is a standard library that contains several functions that are useful in functional programming. One type of function it produces is **infinite itertools**.

The function **count()** counts up infinitely from a value.

The function **cycle()** infinitely iterates through an iterable (for instance, a list or string).

The function **repeat()** repeats an object, either infinitely or a specific number of times.

from itertools import count

for i in count(3):

```
    print(i)
    if i >= 11:
       break
```

Output 3 4 5 6 7 8 9 10 11

Some functions in itertools work in a way similar to map & filters.

takewhile -- takes items from an iterable while a predicate function remains true.

chain -- combines severable iterables into a long one.

accumulate -- returns a running total of values in an iterable.

Turtle Program to Print "Rainbow Benzene"[edit]

import turtle

colors = ['red', 'purple', 'blue', 'green', 'orange', 'yellow']

t = turtle.Pen()

turtle.bgcolor('black')

for x in range(360):

```
   t.pencolor(colors[x%6])
   t.width(x/100+1)
   t.forward(x)
```

Python Imaging Library (PIL)[edit]

Python Imaging Library (abbreviated as PIL) (in newer versions known as Pillow) is a free library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats.

Capabilities

Pillow offers several standard procedures for image manipulation. These include:

- Per-pixel manipulations
- Masking and transparency handling
- Image filtering, such as blurring, contouring, smoothing, or edge finding

- Image enhancing, such as sharpening, adjusting brightness, contrast or color
- Adding text to images and much more

Operations With Images[edit]

- **To open a particular image from a path**

try:

```
   img  = Image.open(path)
```

except IOError:

```
   pass
```

- **To retrieve the size of image**

from PIL import Image

filename = "image.png"

with Image.open(filename) as image:

```
   width, height = image.size
```

- **Save changes in image**

img.save(path, format)

```
#format is optional, if no format is specified,
#it is determined from the filename extension
```

- **Rotating an image**

from PIL import Image

def main():

```
   try:
       #Relative Path
       img = Image.open("picture.jpg")

       #Angle given
       img = img.rotate(180)

        #Saved in the same relative location
       img.save("rotated_picture.jpg")
   except IOError:
       pass
```

```
if __name__ == "__main__":

    main()
```

- **Cropping an image**

```
from PIL import Image

def main():

    try:
        #Relative Path
        img = Image.open("picture.jpg")
        width, height = img.size


        area = (0, 0, width/2, height/2)
        img = img.crop(area)


        #Saved in the same relative location
        img.save("cropped_picture.jpg")


    except IOError:
        pass

if __name__ == "__main__":

    main()
```

- **Creating a thumbnail**

This method creates a thumbnail of the image that is opened. It does not return a new image object, it makes in-place modification to the currently opened image object itself. If you do not want to change the original image object, create a copy and then apply this method. This method also evaluates the appropriate to maintain the aspect ratio of the image according to the size passed.

```
from PIL import Image

def main():

    try:
        #Relative Path
        img = Image.open("picture.jpg")
```

```
        #In-place modification
        img.thumbnail((200, 200))

        img.save("thumb.jpg")
    except IOError:
        pass

if __name__ == "__main__":

    main()
```

Animated Spiral with Colors added[edit]

```
from turtle import *
from math import *
pensize(10)
pencolor("blue")
penup()
radius = 20
goto(radius,0)
pendown()
for i in range(1,300):
 newangle = 2 * i * pi / 100   //indent this
 goto( radius*cos(newangle),
 radius*sin(newangle) )
 radius = radius + 1
```

An Amazing Design With Turtle[edit]

```
import turtle

colors = ['red', 'purple', 'blue', 'green', 'orange', 'yellow']

t = turtle.Pen()

turtle.bgcolor('black')

for x in range(360):

    t.pencolor(colors[x%6])
```

```
t.width(x/100+1)
t.forward(x)
t.left(59)
```

Lecture Notes:Week 8

From JOCWiki

Contents

[hide]

- 1 **SETS IN PYTHON**
  - 1.1 *Creating Sets*
  - 1.2 *Frozen Sets*
  - 1.3 *Adding Elements to a set*
  - 1.4 *UNION of two sets*
  - 1.5 *INTERSECTION of two sets*
  - 1.6 *DIFFERENCE of two sets*
  - 1.7 *REMOVE all elements from a set*
- 2 **Conversion from Decimal Number System to other bases(octal,hexadecimal,binary etc)**
- 3 **DEQUE : Double-ended queue**
  - 3.1 *append(x)*
  - 3.2 *appendleft(x)*
  - 3.3 *clear()*
  - 3.4 *count(x)*
  - 3.5 *extend(iterable)*
  - 3.6 *extendleft(iterable)*
  - 3.7 *pop()*
  - 3.8 *popleft()*
  - 3.9 *remove(value)*
  - 3.10 *reverse()*
- 4 **Program on deque**
- 5 **Tuples**
  - 5.1 Different names of different brackets.
  - 5.2 Tuples in brief
  - 5.3 **Few more points on Tuples**
- 6 **NumPy Array**
- 7 **Upper Triangular Matrix**
- 8 **Lottery Simulation**
- 9 **Lottery Simulation**
- 10 **Image Processing**
- 11 **Anagrams**
- 12 **Facebook sentiment analysis**
- 13 **Another solution to upper triangular matrix**
- 14 **Solution to Binary matrix**
- 15 **Solution to Binary matrix**
- 16 **Solution to whether matrix is symmetric or not**
- 17 **Program for Snakes and Ladders (Advanced Version)**
  - 17.1 *Key Features*

- 18 **BINARY SEARCH**
- 19 CUBES ON ONE ANOTHER! Program
- 20 **Matplotlib**
- 21 **Labeling and styling plot in matplotlib**
- 22 **What is Natural Language Processing ?**
- 23 **What is NLTK ?**
- 24 **Various NLP Libraries**
- 25 **Lexical Analysis**
- 26 **Token**
- 27 **What is Stemming ?**
- 28 **Program to Understand Stemming**
- 29 **What is Lemmatization ?**
- 30 **Why is Lemmatization better than Stemming ?**
- 31 **What is Sentiment Analysis ?**
- 32 **Why is Sentiment Analysis So Important ?**
- 33 **VADER Sentiment Analysis**
- 34 **Code For Vader Sentiment Analysis**
- 35 **Advantages of Using VADER**
- 36 **PANDAS**
- 37 **Key Features of PANDAS**
- 38 **Adaptive Histogram Equalization**
- 39 **What is OpenCV ?**

SETS IN PYTHON[edit]

Posted By: Som

- Set is a data type that is unordered, iterable, and has no duplicate elements.
- Set is better as compared to a list, when we are required to check whether or not an element is present in the set or not.
- Set are quick because they are based on data structures called hash tables.
- Sets in Pyhton are quite similar to the mathematical notion of sets.

Creating Sets[edit]

```
1 my_set = set([1,4,5,6])
2 print(my_set)
3 #OUTPUT {1,4,5,6}
```

```
1 my_set = {1,4,5,6}
2 print(my_set)
3 #OUTPUT {1,4,5,6}
```

```
1 my_set = set({1,4,5,6}) #NOTE the curly braces here
2 print(my_set)
3 #OUTPUT {1,4,5,6}
```

```
my_set = set({1,7,2,6})
print(my_set)
#OUTPUT {1,2,6,7} #Note how the elements get sorted on its own!
```

```
my_set = set({1,7,2,6,6,1})
print(my_set)
#OUTPUT {1,2,6,7} #Note how redundant elements are eliminated
```

Frozen Sets[edit]

```
#Creating Frozen Sets
my_frozenset = frozenset(['a','c','f','e'])
print(my_frozenset)
#Output: frozenset({'f', 'c', 'e', 'a'})
```

NOTE: You cannot add elements into a frozen set

Adding Elements to a set[edit]

```
my_set = set({1,7,2,6})
my_set.add(5)
print(my_set)
#OUTPUT: {1, 2, 5, 6, 7}
```

UNION of two sets [edit]

```
even = set({2,8,4,6})
odd = set({1,9,7,11})
number = even.union(odd)
print(number)
#OUTPUT: {1, 2, 4, 6, 7, 8, 9, 11}
```

```
even = set({2,8,4,6})
odd = set({1,9,7,11})
number = even|odd #Another way
print(number)
#OUTPUT: {1, 2, 4, 6, 7, 8, 9, 11}
```

INTERSECTION of two sets [edit]

```
e = set({1,8,7,6})
```

```
o = set({1,9,7,11})
number = e.intersection(o)
print(number)
#OUTPUT: {1,7}
```

DIFFERENCE of two sets [edit]

```
e = set({1,8,7,6})
o = set({1,9,7,11})
number = e-o
print(number)
#OUTPUT: {8,6}
```

REMOVE all elements from a set [edit]

```
my_set = set({1,8,7,6})
my_set.clear()
print(my_set)
#OUTPUT: set()
```

Conversion from Decimal Number System to other bases(octal,hexadecimal,binary etc) [edit]

Posted By: Som

```
def base_change(number,base):
    import string
    s = []
    n = number
    alpha = string.ascii_uppercase
    while n>=base:
        if n%base <= 9:
            s.append(n%base)
            n = int(n/base)
        else:
            s.append(alpha[(n%base)-10])
            n = int(n/base)
```

```
    if n%base <= 9:
        s.append(n)
    else:
        s.append(alpha[(n%base)-10])
    s.reverse()
    ans = ''
    for i in s:
        ans = ans + str(i)
    return ans
```

```
base_change(9,2)
#OUTPUT '1001'
#Returns a string because base 11 onwards we have strings as symbols also.
#Like Base 16 has characters like A,B,C,D,E & F.
```

```
base_change(79,16)
#OUTPUT '4F'
#Converting 79 in base 10 to its equivalent in base 16
```

DEQUE : Double-ended queue[edit]

Posted By: Som

- *A deque is a double-ended queue. It can be used to add or remove elements from both ends.*
- Deque can be imported from collections library.
- The different methods in deque are listed below

append(x)[edit]

Posted By: Som

- Add x to the right side of the deque.

```
from collections import deque
d = deque()
d.append('apple')
print(d)
# OUTPUT : deque(['apple'])
```

appendleft(x)[edit]

Posted By: Som

- Add x to the left side of the deque.

```
d = deque()
d.append('apple')
d.append('banana')
d.appendleft('guava')
print(d)
#OUTPUT : deque(['guava', 'apple', 'banana'])
```

clear()[edit]

Posted By: Som

- Removes all elements from the deque, leaving it with length zero.

```
d = deque()
d.append('apple')
d.append('banana')
d.appendleft('guava')
print(d)
d.clear()
print(d)
'''
OUTPUT:
deque(['guava', 'apple', 'banana'])
deque([])
'''
```

count(x)[edit]

Posted By: Som

- Counts the number of deque elements equal to x and returns the count.

```
d = deque()
d.append('apple')
d.append('banana')
d.appendleft('apple')
d.append('apple')
d.append('berries')
```

```
7 d.appendleft('banana')
8 print(d.count('apple'))
#OUTPUT: 3
```

extend(iterable)[edit]

Posted By: Som

- Extends the right side of the deque by appending elements from the iterable argument(like lists,tuples).

```
1 d = deque()
2 d.append(2)
3 d.extend([3,6,4])
4 d.extend(('one','five','two'))
5 print(d)
#OUTPUT : deque([2, 3, 6, 4, 'one', 'five', 'two'])
```

extendleft(iterable)[edit]

Posted By: Som

- Extends the left side of the deque by appending elements from iterable.Ex: [1,2,3] is inserted in sequence 3,2,1 from left

```
1 d = deque()
2 d.append('a')
3 d.extendleft([3,6,4])
4 print(d)
#Output: deque([4, 6, 3, 'a'])
```

pop()[edit]

Posted By: Som

- Removes and returns the element present on the right side of the deque. If no element is present, raises an IndexError.

```
1 d = deque()
2 d.append('a')
3 d.extendleft([3,6,4])
4 print(d)
5 d.pop()
'''
```

```
OUTPUT:
deque([4, 6, 3, 'a'])
'a'
'''
```

popleft()[edit]

Posted By: Som

- Removes and returns the leftmost element of the deque. If no element is present, raises an IndexError.

```
1 d = deque()
2 d.append('a')
3 d.extendleft([3,6,4])
4 print(d)
5 d.popleft()
6 print(d)
'''
OUTPUT:
deque([4, 6, 3, 'a'])
deque([6, 3, 'a'])
'''
```

remove(value)[edit]

Posted By: Som

- Removes the first occurrence of the value. If not found, raises a ValueError.

```
1 d = deque()
2 d.append('a')
3 d.extendleft([3,6,'a'])
4 print(d)
5 d.remove('a')
6 print(d)
'''
OUTPUT:
deque(['a', 6, 3, 'a'])
deque([6, 3, 'a'])
'''
```

reverse()[edit]

Posted By: Som

- Reverses the elements of the deque *in-place* and then return None.

```
d = deque()
d.append('a')
d.extendleft([3,6,'b'])
d.extend((70,90,100))
print(d)
d.reverse()
print(d)
'''
OUTPUT:
deque(['b', 6, 3, 'a', 70, 90, 100])
deque([100, 90, 70, 'a', 3, 6, 'b'])
'''
```

Program on deque[edit]

```
from collections import deque
d = deque()
for i in range(int(input())):
    x = list(input().split())
    if x[0] == 'append':
        d.append(int(x[1]))
    elif x[0] == 'pop':
        d.pop()
    elif x[0] == 'popleft':
        d.popleft()
    else:
        d.appendleft(int(x[1]))
for i in d:
    print(i,end=' ')
```

```
'''
SAMPLE INPUT :
6
append 10
append 20
append 30
appendleft 44
pop
popleft
'''
'''
OUTPUT
10 20
'''
```

Tuples[edit]

Posted By: Arunkumar | Doubts on this topic?

Tuples is a interesting Data structure. This is not like any other Data structure we have learnt so far. Once we add data into a tuple, tuple wont allow us to edit the data. That is why we call Tuple as a "immutable". But this is not true all the time. Other than this unique behaviour of Tuple, still we have the indexing feature.

Example: - ***Tuple_Examples1.py***

```
"""
Tuple uses parentheses.
"""
tuple_test=('T','E','S','T')
print(tuple_test)
print(tuple_test[1])

#Output:
#('T','E','S','T')
#'E'
"""
But doing this will through an error, as Tuple is immutable.
```

```
13 Comment below lines to escape error.
14 """
15 tuple_test[4]='not'
16 tuple_test[3]='POSSIBLE'
17
18 #Output:
19 #TypeError: 'tuple' object does not support item assignment
```

As we saw earlier, tuple is immutable. But, we are allowed to have "**mutable objects**" in tuple. *This is where it becomes interesting, yet complicated, as we need to deal with index numbers*. To keep it simple, we can say. Tuple is immutable, yet overwritable, and it can have mutable objects. Let's understand all this with examples.

Example: - *Tuple_Examples2.py*

```
 1 """
 2 Overwrite is allowed on Tuple.
 3 """
 4 tuple_test=('it','is','not','POSSIBLE?')
 5 print(*tuple_test)
 6 tuple_test=('But,','overwrite','is','POSSIBLE!')
 7 print(*tuple_test)
 8
 9 #Output:
10 #it is not POSSIBLE?
11 #But, overwrite is POSSIBLE!
12
13 """
14 Multiple mutable objects are also allowed.
15 Lets create two lists inside Tuple. But remember, once we create
Tuple,
16 we can't add another new object to the tuple.
17 """
18 tuple_test=(['Multiple','mutable'],['objects?'])
19 print('This is list 0 -',tuple_test[0])
20 print('This is list 1 -',tuple_test[1])
21
22 #Output:
```

```
#This is list 0 - ['Multiple', 'mutable']
#This is list 1 - ['objects?']

"""
Lets see if we can append, insert, del etc... on lists now.
If one of these is possible, then everything is possible.
You try the others. I mean, append, del, pop etc...
"""
print(*tuple_test[0],*tuple_test[1])
tuple_test[0].insert(0, 'YES!')
tuple_test[1][0]='objects'
tuple_test[1].append('are possible!')
print(*tuple_test[0],*tuple_test[1])

#Output:
#Multiple mutable objects?
#YES! Multiple mutable objects are possible!
```

Now let's deal with complex things. Is it possible to add different types of objects to a Tuple? Is it possible to add matrix too?

Example: - ***Tuple_Examples3.py***

```
"""
Complexity is just in our mind. To be frank this is not-at-all complex.
But Yes, this will be confusing.
Lets add a matrix and a dictionary to a Tuple.
"""
tuple_test=([[11,12,13],
             [14,15,16],
             [17,18,19]],{})

tuple_test[1]['i']='integer'
print(tuple_test)

tuple_test[1]['f']='float'
```

```
tuple_test[1]['s']='string'
tuple_test[1]['l']='list'
tuple_test[1]['d']='dictionary'
tuple_test[1]['t']='tuple'
print(tuple_test[0])
print(tuple_test[0][1])
print(tuple_test[0][2][3])
print('This is a',tuple_test[1]['d']+'!', 'inside a',tuple_test[1]['t']+'!')

#Output:
#([[11, 12, 13], [14, 15, 16], [17, 18, 19]], {'i': 'integer'})
#[[11, 12, 13], [14, 15, 16], [17, 18, 19]]
#[14, 15, 16]
#19
#This is a dictionary! inside a tuple!
```

Different names of different brackets.[edit]

These are different brackets we use as a programming person.

`[]` - Square brackets or box brackets.

`{}` - Flower brackets, Curly brackets or braces.

`()` - Curve brackets, Round brackets, parens or parentheses.

`<>` - Angled brackets or pointy brackets.

Tuples in brief[edit]

Posted By: Divya98

Tuples is a fixed and a restricted Data Structure which allow us to manage data. A tuple is a sequence of immutable python objects, as we cannot add , modify and delete individual elements of tuples.

Operations on tuples:

Creating a tuple:

Syntax- name of tuples= items in a tuple separated by commas enclosed in a parentheses.

nptel_courses=("JOC","Discrete Mathematics","Cloud Computing","Blockchain")

Accessing values in a tuples: Index of elements in a tuples always begin from 0.

nptel_courses=("JOC","Discrete Mathematics","Cloud Computing","Blockchain")

print(nptel_courses[0]) #nameOfTuple[index]

14. **output - JOC**

This will print element on that particular index

Updating Tuples

Tuples are immutable which means you cannot update or change the values of tuple elements.

nptel_courses[2]="Machine Learning"

15. **Output:**
16. **TypeError: 'tuple' object does not support item assignment.**

Delete Tuple Elements<\b>
Removing individual tuple elements is not possible in tuples.
<b>del nptel_courses[3]

17. **Output:**
18. **TypeError: 'tuple' object does not support item assignment**

As Tuples are immutable you cannot delete an individual element. To explicitly remove an entire tuple, just use the del statement.
del nptel_courses

Few more points on Tuples[edit]

Posted By: TheAvenger

1)Tuple is data structure in python.It is collection of objects.

2)They are immutable i.e. we can't change once defined

3)Tuples have values inside round brackets separated by comma .

4)Python code written by using tuple is little faster than with using list

5)Empty tuple can be created as ->

```
  Hotels=()
```

6)We can create tuple like ->

```
 i)Hotels=('Taj','Pink Villa','blue heights')
```

OR

```
 ii)Hotels='Taj','Pink Villa','blue heights'
           Yes, we can create tuple without using round brackets as
well.
```

7)Indexing in tuple starts from 0 just like in list.

8)To print complete tuple we can do by two ways :

```
i)Hotels
ii)print(Hotels)
```

9)To print each item in tuple individually :

```
for item in Hotels:
    print(item)
output :Taj
```

Pink Villa blue heights

10)To access any item at particular index in tuple .It can be accessed as :

```
print(Hotels[1])
it will print Pink Villa
```

11)Difference between tuple and list is that tuple are immutable and lists are mutable.Another difference is Tuples use parentheses i.e. round brackets, whereas lists use square brackets.

12)tuple containing a single value you have to include a comma, even though there is only one value

```
  e.g. Students=(50,)
```

- Operations can't be performed on tuple :

Item at particular index in tuple can't be deleted or modified.Item can't be added at particular index.

- Operations can be performed on tuple :

```
1)Entire tuple can be deleted by using 'del' keyword
  del Hotels
2)len(Tuple_Name) will return length of tuple :
  e.g. len(Hotels) will print length of tuple Hotels .
3)Tuple_Name.count(item) will return number of times that item
appeared in tuple.
  e.g. print(Hotels.count("Taj"))
       will print 1
4)Important note :Tuples can be redefined even though they can't be
modified .i.e.
  Hotels=('Taj','Pink Villa','blue heights','Lion King','Modern
world') this is allowed.
5)Two tuples can be concatenated as follows
   Fruits=('Apple','Orange')
   RollNumbers=(1,2,3,4)
   print(Fruits+RollNumbers)
   output:('Apple','Orange',1,2,3,4)
6)Creating nested tuples
  Fruits=('Apple','Orange')
  RollNumbers=(1,2,3,4)
  Resultant=(Fruits,RollNumbers)
  print(Resultant)
  output:(('Apple', 'Orange'), (1, 2, 3, 4))
7)
   colour=('Red',)*3
   output:('Red','Red','Red')
   In above example if we don't write comma after item then
   colour=('Red')*3
   output:('Red','Red','Red')
   output:RedRedRed
8)We can do slicing o tuple
  RollNumbers=(1,2,3,4)
  print(RollNumbers[1:])
  print(RollNumbers[::-1])
  print(RollNumbers[2:4])
  output:
  (2, 3, 4)
  (4, 3, 2, 1)
  (3, 4)
9)Converting list into tuple
  Numbers = [ 1, 2,5,6]
  print(tuple(Numbers))
  print(tuple('Avengers'))
  Output:
  (1, 2, 5, 6)
  ('A', 'v', 'e', 'n', 'g', 'e', 'r', 's')
10)Movies = ('Avengers',)
   n = 5
   for i in range(int(n)):
```

Movies = (Movies,) print(Movies)

```
output:
(('Avengers',),)
(('Avengers',),),)
((('Avengers',),),),)
(((('Avengers',),),),),)
((((('Avengers',),),),),),)
```

- Tuples can be used at places were we do not want our items in tuple to get modified.

Posted By: Divya98

1.A tuple is a collection which is ordered and unchangeable or we can say tuples are immutable objects.</br> 2.In Python tuples are written with round brackets.</br> 3.The empty tuple is written as two parentheses containing nothing −

```
nptel_courses()
```

4.To write a tuple containing a single value you have to include a comma, even though there is only one value −

```
nptel_courses("JOC",)
```

5.Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.
6.Tuples respond to the + and * operators.
7.To know the length of tuple you use len function

```
len(nptel_courses)

#output:- 4
```

8.To count the occurence of an individual element , you need to use count function

```
nptel_courses.count("JOC")

#output:- 1
```

9.Built-in Tuple Fuctions:

```
i. cmp(tuple1, tuple2): Compares elements of both tuples.
ii.       len(tuple): Gives the total length of the tuple.
iii.      max(tuple): Returns item from the tuple with max value.
iv.       min(tuple): Returns item from the tuple with min value.
 v.       tuple(seq): Converts a list into tuple.
```

10. To print tuple elements line by line You will use a for loop

```
   nptel_courses=("JOC","Discrete Mathematics","Cloud
Computing","Blockchain")

   for courses in nptel_courses:

   print(courses)
```

11.To determine if a specified item is present in a tuple use the in keyword:

```
  Check if "JOC" is present in the tuple:
```

```
  nptel_courses=("JOC","Discrete Mathematics","Cloud
Computing","Blockchain")

  if "JOC" in nptel_courses:

  print("Yes, 'JOC' is in the nptel_courses tuple")
```

NumPy Array[edit]

Posted By: divya98 NumPy is a package for scientific computing which has support for a powerful N-dimensional array object.
NumPy provides multidimensional array of numbers (which is actually an object).

Let's take an example:

```
import numpy as np

a = np.array([1, 2, 3])

print(a)

print(type(a))
# Output: [1, 2, 3]
```

NumPy's array class is called ndarray.

Upper Triangular Matrix[edit]

Posted By: Som

- An upper triangular matrix is a square matrix in which all the element below main diagonal are zero.
- An upper triangular matrix U is in the form,
  - aij for i<=j
  - 0 for i>j

```
n = int(input())
a=[]
for i in range(n):
  l = []
  for j in range(n):
    l.append(0)
  a.append(l)
for i in range(n):
  a[i] = list(map(int,input().split()))
for i in range(n):
  for j in range(n):
    if(i>j):
      a[i][j] = 0
```

```
for i in range(n):
  for j in range(n):
    if j != n-1:
      print(a[i][j],end=' ')
    else:
      print(a[i][j],end='')
  if i!=n-1:
       print()
```

Lottery Simulation[edit]

1. It is a system of say 'N' units , and you bet on 1 unit. If you win you all the 'N-1' units and if you loose you loose that 1 unit too.
2. Each unit has some amount associated with it. Say all the systems in a unit cost Rs. M .If a person wins he get M*(N-1) rs.But if he looses then he will loose 'M' Rupees invested by him.

More detailed Explanation
1)Basically there are some sort of coupons was available and you haveto bet on one particular coupon and there draw coupon randomly from the available list ofcoupon and if your betted coupon and the coupon that appears matches you will win for playing this game you are supposed to place some amount and if you win you get something much more than what you have paid.

2)As the number of units increases the person has to undergo more loss.

3)As number of days in which he draws lottery increases person has to undergo more loss.

e.g.
Suppose there are 'N' tickets of lottery Let N be 10.And to buy or bet on 1 ticket he has to pay 'M' rupees Let M be 100 rupees.Then if number on his lottery ticket matches to number drawn from lucky draw by organiser then person wins he will earn (N-1)*M i.e 900 rs. If he looses then he also looses amount he paid to buy ticket i.e.100 Rupees therefore 100 Rupees will be deducted from his account.

Suppose ,there are 1000 tickets from which he has to choose then his winning probability will reduce than previous case when he has to choose from 10 tickets only.

Suppose ,He is playing for say a year then in an average he will face more losses as compared to if he plays for month.

Method taught in lectures with little modification

```
import random
import matplotlib.pyplot as plt

account=0
x=[]
```

```
 6 y=[]
 7
 8 Amount_to_be_invested=int(input("Enter amount need to be
invested to buy 1 lottery ticket "))
 9 Total_Number_Of_tickets=int(input("Enter total number of tickets
"))
10 NumberOfDays=int(input("Enter number of days for which you want
to find out amount of money may be earned "))
11 for i in range(NumberOfDays):
12     x.append((i+1))
13     bet=random.randint(1,Total_Number_Of_tickets)
14     Lucky_Draw=random.randint(1,Total_Number_Of_tickets)
15     if(bet==Lucky_Draw):
16         account=account+((Total_Number_Of_tickets-
1)*Amount_to_be_invested)-Amount_to_be_invested
17     else:
18         account=account-Amount_to_be_invested
19     y.append(account)
20
21 print("Amount in your account is:",account)
22 plt.plot(x,y)
23 plt.ylabel("Account Balance")
24 plt.xlabel("Days")
25 plt.show()
```

Lottery Simulation[edit]

Posted By: TheAvenger

1)It is system of 'N' units in which person have to bet on 1 unit and have to invest suppose 'M' Rupees on his bet.In a lucky draw if the number on his unit matches to wining number then he wins.If the person wins he get all 'N-1' units i.e. he will earn (M*(N-1)) Rupees .But if he looses then he will loose 'M' Rupees invested by him .

2)As the number of units increases the person has to undergo more loss.

3)As number of days in which he draws lottery increases person has to undergo more loss.

e.g. Suppose there are 'N' tickets of lottery Let N be 10.And to buy or bet on 1 ticket he has to pay 'M' rupees Let M be 100 rupees.Then if number on his lottery ticket matches to number drawn from

lucky draw by organiser then person wins he will earn (9*100=900) but he has already paid 100 rupees .Therefore amount in his account will be 800 Rupees i.e.(900-100).If he looses then he also looses amount he paid to buy ticket i.e.100 Rupees therefore 100 Rupees will be deducted from his account.

Suppose ,there are 1000 tickets from which he has to choose then his winning probability will reduce than previous case when he has to choose from 10 tickets only.

Suppose ,He is playing for say a year then in an average he will face more losses as compared to if he plays for month.

Method taught in lectures with little modification

```
 1 import random
 2 import matplotlib.pyplot as plt
 3
 4 account=0
 5 x=[]
 6 y=[]
 7
 8 Amount_to_be_invested=int(input("Enter amount need to be
invested to buy 1 lottery ticket "))
 9 Total_Number_Of_tickets=int(input("Enter total number of tickets
"))
10 NumberOfDays=int(input("Enter number of days for which you want
to find out amount of money may be earned "))
11 for i in range(NumberOfDays):
12     x.append((i+1))
13     bet=random.randint(1,Total_Number_Of_tickets)
14     Lucky_Draw=random.randint(1,Total_Number_Of_tickets)
15     if(bet==Lucky_Draw):
16         account=account+((Total_Number_Of_tickets-
1)*Amount_to_be_invested)-Amount_to_be_invested
17     else:
18         account=account-Amount_to_be_invested
19     y.append(account)
20
21 print("Amount in your account is:",account)
22 plt.plot(x,y)
23 plt.ylabel("Account Balance")
24 plt.xlabel("Days")
```

```
plt.show()
```

Image Processing[edit]

Image Processing is used to transforming the image from one to form that can be understood by all.Image processing is done so that we can know the minute features of the images.

PIL:
PIL stands for Python Image Library. It is a free library for the python programming language. It is used for opening, changing or modifying, saving, etc of images. The newer version of PIL is known as Pillow.

Pillow offers several standard procedures for image manipulation. These include:

per-pixel manipulations,
masking and transparency handling,
image filtering, such as blurring, contouring, smoothing, or edge finding,
image enhancing, such as sharpening, adjusting brightness, contrast or color,
adding text to images and much more.
It is available for Windows, Mac OS X and Linux.

We can prerform

- Image enhancing
- Changing contrast or brightness
- Image rotation
- Change color, blur, sharpen

and many more operations. Formats supported by PIL are PPM, PNG, JPEG, GIF, TIFF, and BMP.

Simple Operations on an image

Flipping the image (mirror image)

```
from PIL import Image
#load the image
img=Image.open('joc.png')

#flip the mirror image
transposed_img=img.transpose(Image.FLIP_LEFT_RIGHT)

#saving the image
transposed_img.save('corrected.png')

print('done flipping')
```

Blurring an image

```
from PIL import Image, ImageFilter

# load an image
original = Image.open("nature.png")

# blur the image
blurred = original.filter(ImageFilter.BLUR)

# display both images
original.show()
blurred.show()
```

Rotating the image by some degrees

```
from PIL import Image

img = Image.open("nature.jpg")

#rotate the photo by 45 degrees
img.rotate(45).show()
```

Converting an image to black and white

```
from PIL import Image
colorimg = Image.open("era.png")
bw = colorimg.convert('L')
bw.show()
```

Creating an image

```
from PIL import Image
img = Image.new(mode='RGB', size=(100,100), color='blue')#this
will create new image of 100x100 pixels
```

```
3 img.save('blue.jpg')
```

Anagrams[edit]

Posted By: TheAvenger

Anagram is a form of word play in which letters of a word or phrase are rearranged in such a way that a new word or phrase is formed.An anagram is formed by using exactly the same letters of the original word, but with a different arrangement.

e.g. car and arc

```
    stew and west
    desserts and stressed
```

- ASCII Values :

1)ASCII stands for American Standard Code for Information Interchange 2)ASCII values represent character encoding 3)It is a code for representing 128 English characters as numbers, with each letter assigned a number from 0 to 127. 4)For following character ascii values are as shown :

```
A-65
Z-90
a-97
z-122
for special symbol '!' ascii value is 33
```

5)print(ord('A'))

```
It will print 65 i.e. ascii value of A
```

6)print(ord('!'))

```
 It will print 33 i.e. ascii value of !
```

- ord()-it returns ascii value for character as well as special symbols

Method 1 :Using sorted() function It is approach taught in video lectures

```
1 FirstString=input("Enter string 1 :")
2 SecondString=input("Enter string 2 :")
3 if(sorted(FirstString)==sorted(SecondString)):
4     print("Given Strings are anagrams")
5 else:
6     print("Given strings are not anagrams")
7 '''OUTPUT
8 Enter string 1 :stressed
9
10 Enter string 2 :desserts
11 Given Strings are anagrams
12 '''
```

Method 2 : Without using inbuild sorted() function

```
FirstString=input("Enter string 1 :")
SecondString=input("Enter string 2 :")
flag=[]                                  #It will just tell whether character we are considering is already considered
count=0

for j in range(len(SecondString)):
    flag.append(0)                       #initializing flag to 0
if(len(FirstString)==len(SecondString)):
    for i in range(len(FirstString)):
        for j in range(len(SecondString)):
            if(FirstString[i]==SecondString[j] and flag[j]!=1):
                count=count+1
                indicator=1
                break


else:
    print("Lengths of Given string are not same therefore ",end="")
if(count==len(SecondString)):
    print("Given Strings are anagrams")
else:
     print("Given strings are not anagrams")
'''OUTPUT
Enter string 1 :dog

Enter string 2 :gods
Lengths of Given string are not same therefore Given strings are not anagrams

Enter string 1 :stressed

```

```
Enter string 2 :desserts
Given Strings are anagrams
'''
```

Method 3

```
FirstString=input("Enter string 1 :")
SecondString=input("Enter string 2 :")          #accepting strings
flag=0
for i in range(256):
    count1=0
    count2=0
    for j in range(len(FirstString)):
        if(i==ord(FirstString[j])):
            count1=count1+1
    for j in range(len(SecondString)):
        if(i==ord(SecondString[j])):
            count2=count2+1
    if(count2!=count1):
        flag=1
        '''if flag set to 1 inbetween then it indicates that in given two strings
           one or more characters are not same times appearing
        '''
if(flag==0):
    print("Given Strings are anagrams")
else:
    print("Given strings are not anagrams")
'''
OUTPUT
Enter string 1 :fried
Enter string 2 :fired
Given Strings are anagrams


```

```
29
30 Enter string 1 :iron
31 Enter string 2 :trol
32 Given strings are not anagrams
33 '''
```

Facebook sentiment analysis[edit]

Posted By: TheAvenger

Sentiment analysis helps to figure out whether particular text is positive ,negative or neutral. for performing facebook sentiment analysis following steps should be done :

1)We need to download facebook data :

```
   i)Log in to your facebook account
   ii)go to settings then click on 'your facebook information'
   iii)Then click on 'Create file' button .
```

2)After downloading facebook information .Store that information in Excel file

3)import pandas library .It provides easy to use data structures

4)import nltk library.It is used to process human language.

5)We will use VADER It is Valence Aware Dictionary and sEntiment Reasoner.It also takes into account the intensity of sentiment.

6)Download VADER lexicon.Lexicon acts as dictionary here.

7)Convert the Excel sheet data with the help of pandas

- Data Frame:

```
A data frame is a 2 Dimensional structure in form of table.
```

Code for sentiment analysis of facebook data

```
1 import pandas as pd
2 import nltk
3 from nltk.sentiment.vader import SentimentIntensityAnalyzer
4
5 nltk.downloader.download('vader_lexicon')      #To download
lexicon type following command
6
7 file='data_file.xslsx'                 #Provide exact path where
data_file.xslx file
```

```
 8 xl=pd.ExcelFile(file)                    #Pandas library helps to
read excel file
 9
10 '''
11 This Excel file we need to pass and convert it to data frames
12 pandas library helps to convert Excel file to data structure
13 which are easy to analyze
14 '''
15
16 dfs=xl.parse(xl.sheet_names[0])           #parsing Excel sheet to
data frame
17 dfs=list(dfs['Timeline'])                #To remove blank rows
from data frames
18 print(dfs)
19
20 sid=SentimentIntensityAnalyzer
21 str1="UTC+05:30"                          # this may be different
for different data
22 for data in dfs:
23  a=data.find(str1)
24  if(a==-1):                                      #If a is -1 then
str1 is not found
25         ss=sid.polarity_scores(data)
26         print(data)
27         for k in ss:
28                 print(k,ss[k])                  #Here k denotes
positive ,negative ,neutral
29                                                          #Here
ss[k] denotes intensity of positive ,negative ,neutral sentiment
```

Pandas Library

pandas is an open source.

Pandas is better at automating data processing tasks than Excel, including processing Excel files.

Pandas DataFrame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes A Data frame is a two-dimensional data structure, data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three components -> data,rows,columns.

DataFrame functions and there description :

1]read_excel : method read the data from the Excel file into a pandas DataFrame object.

2]index() : It will returns index of the DataFrame

3]insert() : This function will insert a column into a DataFrame

4]unique() : this function will extracts the unique values in the dataframe

Pandas is used for routine data analysis tasks as mentioned below:

1]quick Exploratory Data Analysis (EDA)

2]taking cleaned and processed data to any number of data tools

3]building machine learning models.

How to find path of file using command prompt

In linux to find complete path of file you can use command 'locate -br file_name.extension'.
In many programs you will require to provide complete path of your file if it is not present in current working directory.

VADER:
VADER stands for Valence Aware Dictionary and Sentiment Reasoner.With the help of vader we can check how positive or negative a sentence is. We can do this with the help of SentimentIntensityAnalyzer.

Program:

```
1 from vaderSentiment.vaderSentiment import
SentimentIntensityAnalyzer
2 analyzer = SentimentIntensityAnalyzer()
3 sentence = "python is amazing programming language"
4 scores = analyzer.polarity_scores(sentence)
5 print(scores)
```

Output: {'neg': 0.0, 'neu': 0.192, 'pos': 0.808, 'compound': 0.6369}
(Note: 'neg' is negative, 'neu' is neutral and 'pos' is positive.)

Another solution to upper triangular matrix[edit]

Posted By: TheAvenger

Input :Number of rows or coloumns

```
    on next line accept matrix
```

Here we need to convert given matrix to upper triangular matrix(All elements below diagonal are 0)

```
N=int(input())
list1=[]

for i in range(N):
 list1.append([int(i) for i in input().split(" ")])

for i in range(N):
 for j in range(N):
        if(list1[i][j]!=0 and i>j):
                list1[i][j]=0

for i in range(N):
 for j in range(N):
        if(j<(N-1)):
                print(list1[i][j],end=' ')
        else:
                print(list1[i][j],end='')
 if(i<(N-1)):
        print()
'''
INPUT :

3
1 2 3
4 5 6
7 8 9

OUTPUT :

1 2 3
0 5 6
0 0 9
'''
```

Solution to Binary matrix[edit]

Posted By: TheAvenger

```
N,M=map(int,input().split())
list1=[]
count=0

for i in range(N):
 list1.append([int(i) for i in input().split(" ")])

for i in range(N):
 for j in range(M):
        if(list1[i][j]==0 or list1[i][j]==1):
                count=count+1
flag=0

if(count==(M*N)):
 print("YES",end='')
else:
 print("NO",end='')

'''
OUTPUT :
3 4

1 2 1 0

1 0 1 1

0 0 0 1
NO


3 3

1 0 1
```

```
34
35 0 1 0
36
37 0 0 1
38 YES
39 '''
```

Solution to Binary matrix[edit]

Posted By: divya98

```
1 n,m=(input().split(" "))
2 n=int(n)
3 m=int(m)
4 a=[]
5 z=[]
6 for i in range(n):
7     l=[]
8     for j in range(m):
9         l.append(0)
10    a.append(l)
11
12 for i in range(n):
13    o=[int(g) for g in input().split(" ")]
14    for j in range(m):
15        a[i][j]=o[j]
16        z.append(a[i][j])
17
18 for i in z:
19   if i>1:
20     print("NO, it is not a binary matrix",end='')
21     break
22 else:
23   print("YES, it is a binary matrix",end='')
```

Solution to whether matrix is symmetric or not[edit]

Posted By: TheAvenger

```
N=int(input())
list1=[]
count=0;
for i in range(N):
 list1.append([int(i) for i in input().split(" ")])

for i in range(N):
 for j in range(N):
        if(list1[i][j]!=list1[j][i]):
                count=count+1

if(count!=0):
 print("NO",end='')
else:
 print("YES",end='')

'''
OUTPUT:
3

1 2 3

4 5 6

3 2 1
NO


2

1 2

2 1
YES
'''
```

Program for Snakes and Ladders (Advanced Version) [edit]

Posted By: Som

```
import random
snakes = {49:4,30:21,31:19,17:1}
ladders = {3:15,9:20,11:45,12:34}
start1 = False
start2 = False
count1 = 0
count2 = 0
turn = 0
p1 = input("Player 1 Enter your name ")
p2 = input("Player 2 Enter your name ")
while count1 != 50 and count2 != 50 :
    if turn%2 == 0:
        if start1 == False:
            dice = random.randint(1,6)
            print(p1,"your Dice value is",dice)
            if dice == 6:
                print(p1,'you have opened the game')
                start1 = True
            else:
                turn += 1
        while count1!=50 and start1 and turn%2 == 0:
            dice = random.randint(1,6)
            print(p1,"Your Dice value is",dice)
            if count1 + dice <= 50:
                count1 = count1 + dice
                print(p1,"You are at",count1)
                if count1 in snakes: #Matches index
                    count1 = snakes[count1]
                    print(p1,"Snake Bite, you are at",count1)
                if count1 in ladders:
                    count1 = ladders[count1]
```

```
32                      print(p1,"Ladder climbed, you are
at",count1)
33                  if count1 == 50:
34                      print(p1,"You have WON!")
35                      break
36                  if dice == 6:
37                      print("Another throw for",p1)
38                  else:
39                      turn += 1
40              else:
41                  turn += 1
42      if turn%2 == 1:
43          if start2 == False:
44              dice = random.randint(1,6)
45              print(p2,"your Dice value is",dice)
46              if dice == 6:
47                  print(p2,'you have opened the game')
48                  start2 = True
49              else:
50                  turn += 1
51          while count2!=50 and start2 and turn%2 == 1:
52              dice = random.randint(1,6)
53              print(p2,"Your Dice value is",dice)
54              if count2 + dice <= 50:
55                  count2 = count2 + dice
56                  print(p2,"You are at",count2)
57                  if count2 in snakes: #Matches index
58                      count2 = snakes[count2]
59                      print(p2,"Snake Bite, you are at",count2)
60                  if count2 in ladders:
61                      count2 = ladders[count2]
62                      print(p2,"Ladder climbed, you are
at",count2)
63                  if count2 == 50:
64                      print(p2,"You have WON!")
65                      break
```

```
66                  if dice == 6:
67                      print("Another throw for",p2)
68                  else:
69                      turn += 1
70              else:
71                  turn += 1
```

Key Features [edit]

- *Player opens only with a six.*
- *If Player rolls a six, then another throw is given.*
- *If Player is at 49 and rolls up anything other than 1, then player has to wait.*

BINARY SEARCH [edit]

```
1 def binsr(arr,ele):
2     arr.sort()
3     n1=0
4     n2=len(arr)-1
5     while(n1<=n2):
6         mid=int((n1+n2)/2)
7         if ele > arr[mid]:
8             n1=mid+1
9 
10        elif ele < arr[mid]:
11            n2=mid-1
12 
13        else:
14            print("Element Found at pos",mid)
15            return
16 
17    print("Element not Found!")
```

Note: The *FLAG* variable is actually not required, *RETURN* alone does the job.

- Binary search is better as compared to Linear Search.
- Time Complexity of binary search is O(log n) whereas Time Complexity of Linear Search is O(n).

- Binary Search after every iteration reduces the search space by half.

Another Program for Binary Search

```
def binary_search(arr,x):
    arr.sort()
    mid=int(len(arr)/2)
    while True:
        if x<arr[mid]:
            arr = arr[:mid] #Slicing out the lower half
        elif x>arr[mid]:
            arr = arr[mid:] #Slicing out the upper half
        else:
            print("Element Found")
            break
        if mid==0: #To exit the While Loop
            print("Element Not Found")
            break
        mid=int(len(arr)/2)
```

CUBES ON ONE ANOTHER! Program[edit]

Posted By: Som

```
q = int(input())
for i in range(q):
    n = int(input())
    length = list(map(int,input().split()))
    if length[0]>length[-1]:
        array = [length[0]]
    else:
        array=length[-1]
    j = 1; k = 1
    while len(array) != n:
        if array[-1] >= length[j] and array[-1] >= length[-k]:
            if length[j] >= length[-k]:
```

```
                array.append(length[j])
                j += 1
            else:
                array.append(length[-k])
                k+=1
        elif array[-1] >= length[j]:
            array.append(length[j])
            j+=1
        elif array[-1] >= length[-k]:
            array.append(length[-k])
            k+=1

        else:
            break

    if len(array) == n:
        print('Yes')
    else:
        print('No')
```

Matplotlib[edit]

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002. ---- One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc. ---- Windows, Linux and macOS distributions have matplotlib and most of its dependencies as wheel packages. Run the following command to install matplotlib package : ---- "'python -mpip install -U matplotlib"' **Basic plots in Matplotlib** : Matplotlib comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations. They're typically instruments for reasoning about quantitative information. Some of the sample plots are covered here.

Line plot

```
# importing matplotlib module
from matplotlib import pyplot as plt

# x-axis values
x = [5, 2, 9, 4, 7]

# Y-axis values
y = [10, 5, 8, 4, 2]

# Function to plot
```

```
plt.plot(x,y)

# function to show the plot
plt.show()
```

Bar plot

```
# importing matplotlib module
from matplotlib import pyplot as plt

# x-axis values
x = [5, 2, 9, 4, 7]

# Y-axis values
y = [10, 5, 8, 4, 2]

# Function to plot the bar
plt.bar(x,y)

# function to show the plot
plt.show()
```

Histogram

```
# importing matplotlib module
from matplotlib import pyplot as plt

# Y-axis values
y = [10, 5, 8, 4, 2]

# Function to plot histogram
plt.hist(y)

# Function to show the plot
plt.show()
```

Scatter plot

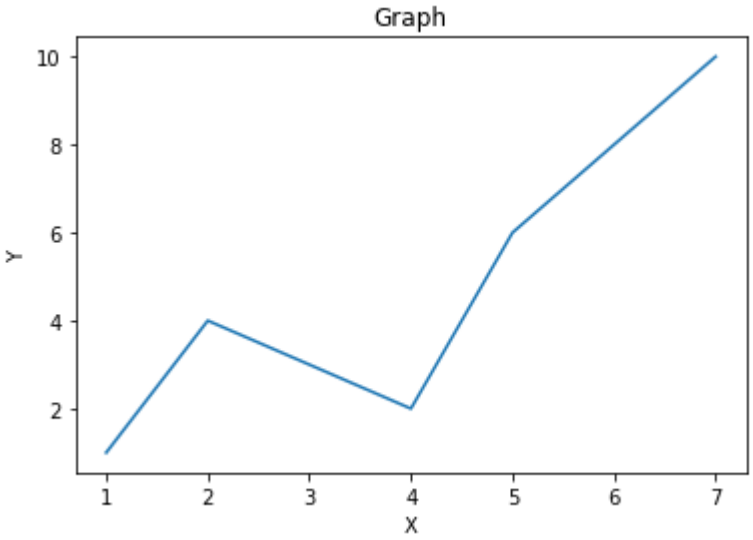
```
# importing matplotlib module
from matplotlib import pyplot as plt

# x-axis values
x = [5, 2, 9, 4, 7]

# Y-axis values
y = [10, 5, 8, 4, 2]

# Function to plot scatter
plt.scatter(x, y)

# function to show the plot
plt.show()
```

Matplotlib requires the following dependencies: • Python (>= 3.5) • FreeType (>= 2.3) • libpng (>= 1.2) • NumPy (>= 1.10.0) • setuptools • cycler (>= 0.10.0) • dateutil (>= 2.1) • kiwisolver (>= 1.0.0) • pyparsing

Labeling and styling plot in matplotlib[edit]

Labeling:

```
from matplotlib import pyplot as plt

# x-axis values
x = [1, 2, 4, 5, 7]

# Y-axis values
y = [1, 4, 2, 6, 10]
plt.plot(x,y)


#labelling the graph
plt.xlabel('X') #xlabel
plt.ylabel('Y') #ylabel
plt.title('Graph')  #title
plt.show()
```

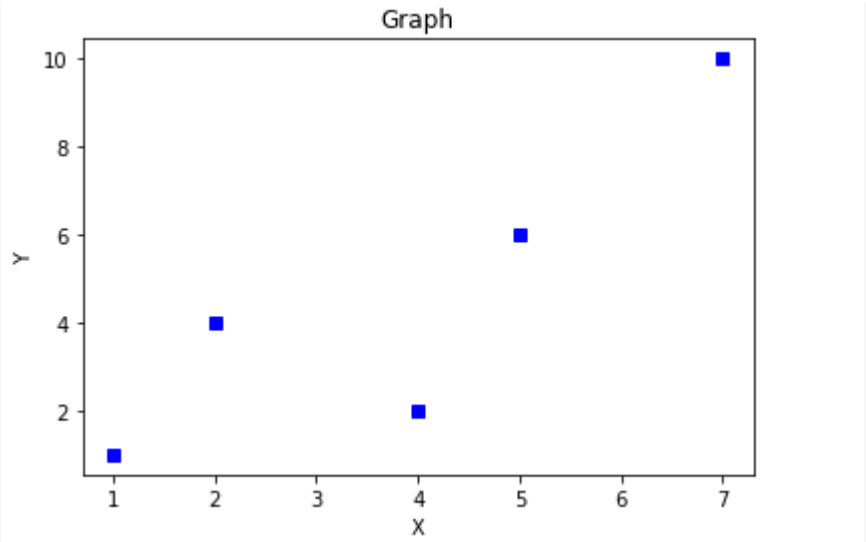


Different chart styles:

1) Blue Squares

In order to display blue squares instead of a line, we can pass 'bs' to plot function.
Example:

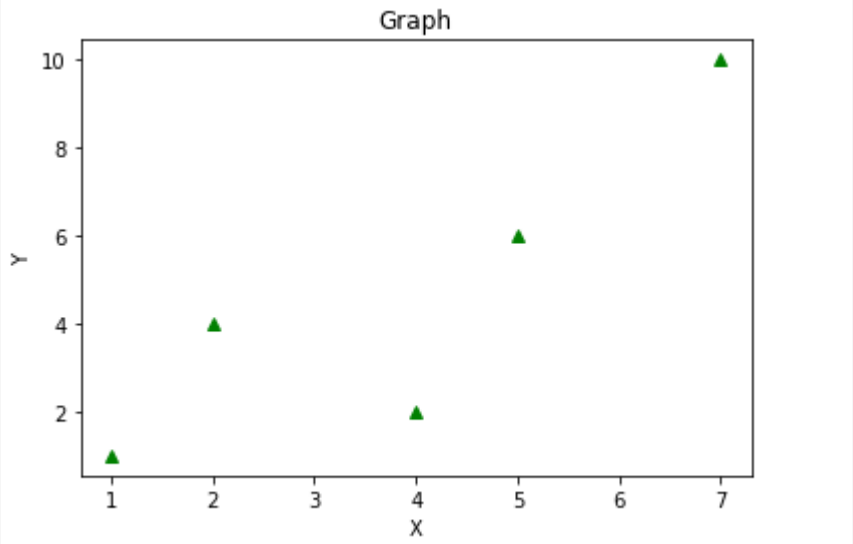
```
# x-axis values
x = [1, 2, 4, 5, 7]

# Y-axis values
y = [1, 4, 2, 6, 10]
plt.plot(x,y,'bs')

#labelling the graph
plt.xlabel('X') #xlabel
plt.ylabel('Y') #ylabel
plt.title('Graph')  #title
plt.show()
```



2) Green triangles

We need to pass 'g^' to plot function.

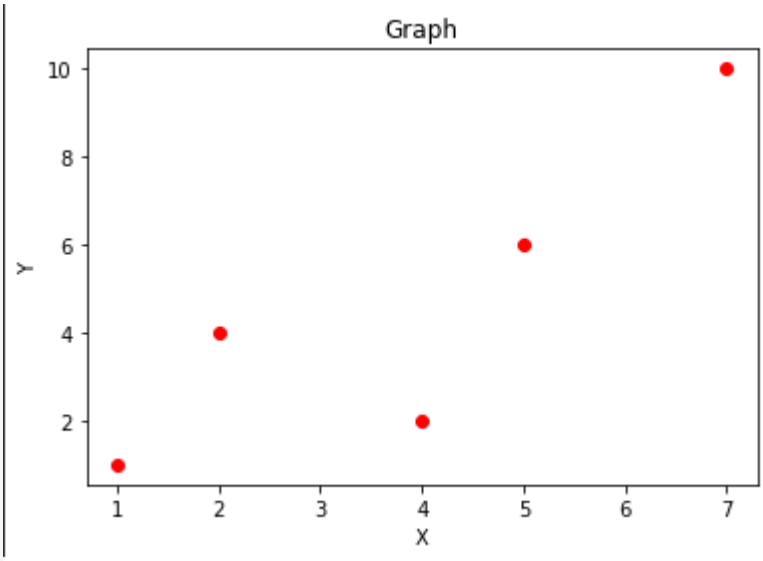
For example: In the above example, we can write plt.plot(x,y,'g^')



3) Red Circles

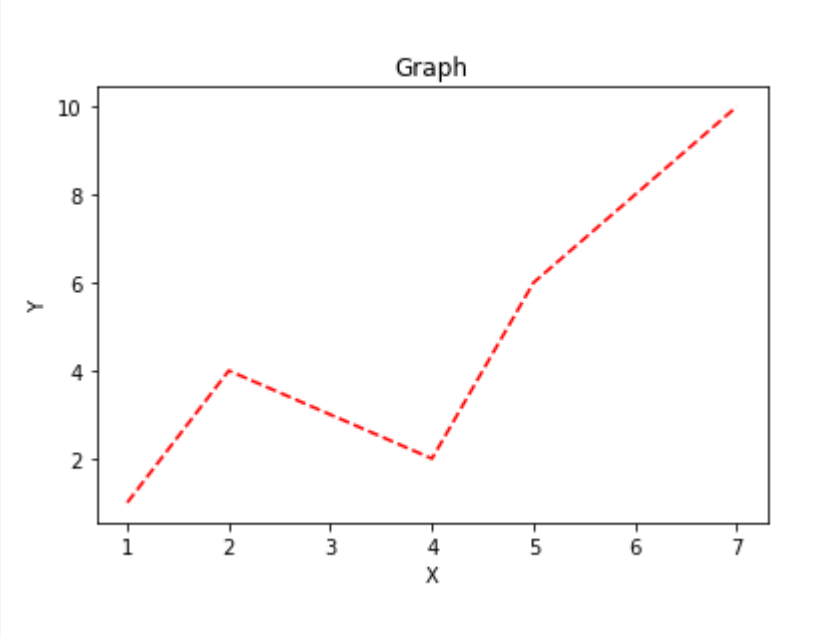
We need to pass 'ro' to plot function.

For example: In the above example, we can write plt.plot(x,y,'ro')



4) Red Dashes

We need to pass 'r--' to plot function.
For example: In the above example, we can write plt.plot(x,y,'r--')



What is Natural Language Processing ?[edit]

Natural language processing (NLP) is about developing applications and services that are able to understand human languages. Some Practical examples of NLP are speech recognition.
For eg: google voice search, understanding what the content is about or sentiment analysis etc.

What is NLTK ?[edit]

Posted By: divya98 NLTK stands for Natural Language Toolkit. The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP).
It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning.

Install NLTK
Install NLTK with Python 2.x using:
sudo pip install nltk

Install NLTK with Python 3.x using:
sudo pip3 install nltk

Installation is not complete after these commands. Open python and type:

```
import nltk
nltk.download()
```

A graphical interface will be presented:

Click all and then click download. It will download all the required packages which may take a while, the bar on the bottom shows the progress.

Tokenize Words
A sentence or data can be split into words using the method word_tokenize():

```
from nltk.tokenize import sent_tokenize, word_tokenize
data = "All work and no play makes jack a dull boy, all work and no play"
print(word_tokenize(data))
```

This will output:
['All', 'work', 'and', 'no', 'play', 'makes', 'jack', 'dull', 'boy', ',', 'all', 'work', 'and', 'no', 'play']

NLTK Stop Words:

Natural language processing (nlp) is a research field that presents many challenges such as natural language understanding
.

Text may contain stop words like ‘the’, ‘is’, ‘are’. Stop words can be filtered from the text to be processed. There is no universal list of stop words in nlp research, however the nltk module contains a list of stop words.

NLTK and arrays If you wish to you can store the words and sentences in arrays:

```
from nltk.tokenize import sent_tokenize, word_tokenize

data = "All work and no play makes jack dull boy. All work and no play makes jack a dull boy."

phrases = sent_tokenize(data)
words = word_tokenize(data)

print(phrases)
print(words)
```

Natural Language Processing -- Prediction:

We can use natural language processing to make predictions.

Example: Given a product review, a computer can predict if its positive or negative based on the text.

Sentiment Analysis:

In Natural Language Processing there is a concept known as Sentiment Analysis.

Given a movie review or a tweet, it can be automatically classified in categories. These categories can be user defined (positive, negative) or whichever classes you want.

Various NLP Libraries[edit]

- **NLTK** -- This is one of the most usable & mother of all NLP libraries.Supporting tasks such as classification, tokenization, stemming, tagging, parsing, and semantic reasoning, this library is your main tool for natural language processing. Today it serves as an educational foundation for Python developers who are dipping their toes in NLP.
- **spaCy** -- This relatively young library was designed for production usage – that's why it's so much more accessible than NLTK. spaCy offers the fastest syntactic parser available on the market today. Moreover, since the toolkit is written in Cython it's also really speedy. In comparison to the libraries we covered so far, it supports the smallest number of languages (seven). However, its growing popularity means that it might start supporting more of them soon.
- **StanfordCoreNLPPython** -- For client-server based architecture, this is a good library in NLTK. It is written in JAVA, but it provides modularity to use it in Python.The library is really fast and works well in product development environments. Moreover, some of CoreNLP components can be integrated with NLTK which is bound to boost the efficiency of the latter.
- **TextBlob** --TextBlob is a must for developers who are starting their journey with NLP in Python and want to make the most of their first encounter with NLTK. It basically provides beginners with an easy interface to help them learn most basic NLP tasks like sentiment analysis, pos-tagging, or noun phrase extraction. It works in Python2 & Python3. This is used for processing textual data & providing mainly all type of operation in the form of API.
- **Genism** -- Gensim is a Python library that specializes in identifying semantic similarity between two documents through vector space modeling and topic modeling toolkit. It can handle large text collections with the help of efficiency data streaming and incremental algorithms, which is more than we can say about other packages that only target batch and in-memory processing.It is a robust open source NLP library support in Python. This is highly efficient & scalable.
- **Pattern** -- It is a light-weighted NLP module. This is generally used in web-mining, crawling or such type of spidering task.
- **Polyglot** --This slightly lesser-known library is one of our favorites because it offers a broad range of analysis and impressive language coverage. For massive multilingual applications, Polyglot is best suitable NLP library. Feature extraction in the way on identity & entity.
- **PyNLPI** -- PyNLPI also was known as "Pineapple" & supports Python. It provides a parser for many data formats like FoLiA/ Giza/ Moses/ ARPA/ Timbl/ CQL.
- **Vocabulary** -- This library is best to get semantic type information from the given text.

Lexical Analysis[edit]

`Posted By: divya98`

Lexical analysis, lexing or tokenization is the process of converting a sequence of characters (such as in a computer program or web page) into a sequence of tokens (strings with an assigned and thus identified meaning).

Lexical analysis is the first phase of a compiler. It takes the modified source code from language preprocessors that are written in the form of sentences. The lexical analyzer breaks these syntaxes into a series of tokens, by removing any whitespace or comments in the source code.

If the lexical analyzer finds a token invalid, it generates an error. The lexical analyzer works closely with the syntax analyzer. It reads character streams from the source code, checks for legal tokens, and passes the data to the syntax analyzer when it demands.

A program that performs lexical analysis may be termed a lexer, tokenizer, or scanner, though scanner is also a term for the first stage of a lexer. A lexer is generally combined with a parser, which together analyze the syntax of programming languages, web pages, and so forth.

Lexeme or Tokens

A lexeme is a sequence of characters in the source program that matches the pattern for a token and is identified by the lexical analyzer as an instance of that token.

The word lexeme in computer science is defined differently than lexeme in linguistics. A lexeme in computer science roughly corresponds to what might be termed a word in linguistics (the term word in computer science has a different meaning than word in linguistics), although in some cases it may be more similar to a morpheme.

Specification of Tokens
Alphabets
Any finite set of symbols {0,1} is a set of binary alphabets, {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F} is a set of Hexadecimal alphabets, {a-z, A-Z} is a set of English language alphabets.

Strings
Any finite sequence of alphabets is called a string. Length of the string is the total number of occurrence of alphabets, e.g., the length of the string nptelcourses is 12 and is denoted by |nptelcourses| = 12. A string having no alphabets, i.e. a string of zero length is known as an empty string and is denoted by ε (epsilon).

Token[edit]

Posted By: divya98

A lexical token or simply token is a string with an assigned and thus identified meaning. It is structured as a pair consisting of a token name and an optional token value. The token name is a category of lexical unit. A token is the basic component of source code . Characters are categorized as one of five classes of tokens that describe their functions (constants, identifiers, operators, key words, and separators) in accordance with the rules of the programming language.

Common token names are:

- **identifier**: Identifier refers to name given to entities such as variables, functions, structures etc. Identifier must be unique. They are created to give unique name to a entity to identify it during the execution of the program

- **keyword**: Keywords are predefined, reserved words used in programming that have special meanings to the compiler. Keywords are part of the syntax and they cannot be used as an identifier.
- **separator (also known as punctuators)**: punctuation characters and paired-delimiters.A delimiter is a sequence of one or more characters used to specify the boundary between separate, independent regions in plain text or other data streams. An example of a delimiter is the comma character, which acts as a field delimiter in a sequence of comma-separated values.

- **operator**: symbols that operate on arguments and produce results.

Various Operators are
Arithmetic Operators.
Relational Operators.
Logical Operators.
Assignment Operators.
Increment and Decrement Operators.
Conditional Operator.
Bitwise Operators.
Special Operators.

- **literal**: numeric, logical, textual, reference literals.
- **comment**: a comment is a programmer-readable explanation or annotation in the source code of a computer program. They are added with the purpose of making the source code easier for humans to understand, and are generally ignored by compilers and interpreters.

Tokenization:

Tokenization is the process of demarcating and possibly classifying sections of a string of input characters. The resulting tokens are then passed on to some other form of processing. The process can be considered a sub-task of parsing input.

What is Stemming ?[edit]

Stemming is a kind of normalization for words. Normalization is a technique where a set of words in a sentence are converted into a sequence to shorten its lookup. The words which have the same meaning but have some variation according to the context or sentence are normalized.

The idea of stemming is a sort of normalizing method. Many variations of words carry the same meaning, other than when tense is involved.

In another word, there is one root word, but there are many variations of the same words. For example, the root word is "eat" and it's variations are "eats, eating, eaten and like so". In the same way, with the help of Stemming, we can find the root word of any variations. The reason why we stem is to shorten the lookup, and normalize sentences.

For example:

- I was taking a ride in the car.
- I was riding in the car.

Both the sentences have the same meaning. in the car is the same. I was is the same. the ing denotes a clear past-tense in both cases, so is it truly necessary to differentiate between ride and riding, in the case of just trying to figure out the meaning of what this past-tense activity was?

No,Not really.

In case we do not provide the same data-set, then machine fails to predict. So it is necessary to differentiate the meaning of each word to prepare the dataset for machine learning. And here stemming is used to categorize the same type of data by getting its root word.

Errors in Stemming:

There are mainly two errors in stemming – **Overstemming** and **Understemming**.

- Overstemming occurs when two words are stemmed to same root that are of different stems.
- Under-stemming occurs when two words are stemmed to same root that are not of different stems.

Applications of stemming are:

- Stemming is used in information retrieval systems like search engines.
- It is used to determine domain vocabularies in domain analysis.
- Stemming is desirable as it may reduce redundancy as most of the time the word stem and their inflected/derived words mean the same.

Some Stemming algorithms are:

Potter's Stemmer algorithm

It is one of the most popular stemming methods proposed in 1980. It is based on the idea that the suffixes in the English language are made up of a combination of smaller and simpler suffixes.
Example: EED -> EE means "if the word has at least one vowel and consonant plus EED ending, change the ending to EE" as 'agreed' becomes 'agree'.

Lovins Stemmer

It is proposed by Lovins in 1968, that removes the longest suffix from a word then word is recoded to convert this stem into valid words.
Example: sitting -> sitt -> sit

Dawson Stemmer

It is extension of Lovins stemmer in which suffixes are stored in the reversed order indexed by their length and last letter.

Krovetz Stemmer

It was proposed in 1993 by Robert Krovetz. Following are the steps:
1) Convert the plural form of a word to its singular form.
2) Convert the past tense of a word to its present tense and remove the suffix 'ing'.
Example: 'children' -> 'child'

Program to Understand Stemming[edit]

```
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
ps = PorterStemmer()
example_words =
["python","pythoner","pythoning","pythoned","pythonly"
for w in example_words:
    print(ps.stem(w))
```

Output:

```
python
python
python
python
python
```

Code Explanation:

- There is a stem module in NLTk which is imported. If you import the complete module, then the program becomes heavy as it contains thousands of lines of codes. So from the entire stem module, we only imported "PorterStemmer."
- There is a tokenize module in NLTK which is imported. so we have import sent_tokenize and word_tokenize.
- We prepared a dummy list 'example words' of variation data of the same word.
- An object is created which belongs to class nltk.stem.porter.PorterStemmer.

- Further, we passed it to PorterStemmer one by one using "for" loop. Finally, we got output example words of each word mentioned in the list.
- From the above explanation, it can also be concluded that stemming is considered as an important preprocessing step because it removed redundancy in the data and variations in the same word. As a result, data is filtered which will help in better machine training.

What is Lemmatization ?[edit]

Lemmatization is the process of grouping together the different inflected forms of a word so they can be analysed as a single item. Lemmatization usually refers to the morphological analysis of words, which aims to remove inflectional endings.

It helps in returning the base or dictionary form of a word, which is known as the lemma. The NLTK Lemmatization method is based on WorldNet's built-in morph function. Text preprocessing includes both stemming as well as lemmatization. Applications of lemmatization are:

- Used in comprehensive retrieval systems like search engines.
- Used in compact indexing

Why is Lemmatization better than Stemming ?[edit]

The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form.

However, the two words differ in their flavor. Stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes. Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma .

Text preprocessing includes both Stemming as well as Lemmatization. Many times people find these two terms confusing. Some treat these two as same. Actually, lemmatization is preferred over Stemming because lemmatization does morphological analysis of the words.

Code for Lemmatization and Stemming

Stemming code

```
import nltk
from nltk.stem.porter import PorterStemmer
porter_stemmer  = PorterStemmer()
text = "studies studying cries cry"
tokenization = nltk.word_tokenize(text)
for w in tokenization:
print("Stemming for {} is {}".format(w,porter_stemmer.stem(w)))
```

Output:

```
Stemming for studies is studi
Stemming for studying is studi
Stemming for cries is cri
Stemming for cry is cri
```

Lemmatization code

```
import nltk
    from nltk.stem import        WordNetLemmatizer
    wordnet_lemmatizer = WordNetLemmatizer()
    text = "studies studying cries cry"
    tokenization = nltk.word_tokenize(text)
     for w in tokenization:
           print("Lemma for {} is {}".format(w,
wordnet_lemmatizer.lemmatize(w)))
```

Output:

```
Lemma for studies is study
Lemma for studying is studying
Lemma for cries is cry
Lemma for cry is cry
```

Discussion of output:

If you look stemming for studies and studying, output is same (studi) but lemmatizer provides different lemma for both tokens study for studies and studying for studying. So when we need to make feature set to train machine, it would be great if lemmatization is preferred.

What is Sentiment Analysis ?[edit]

Sentiment Analysis is the process of 'computationally' determining whether a piece of writing is positive, negative or neutral. It's also known as opinion mining, deriving the opinion or attitude of a speaker.

Sentiment Analysis, or Opinion Mining, is a sub-field of Natural Language Processing (NLP) that tries to identify and extract opinions within a given text. The aim of sentiment analysis is to gauge the attitude, sentiments, evaluations, attitudes and emotions of a speaker/writer based on the computational treatment of subjectivity in a text.

Why is Sentiment Analysis So Important ?[edit]

```
In political field, it is used to keep track of political view, to
detect consistency and inconsistency between statements and actions
at the government level. It can be used to predict election results
as well.
```

Businesses today are heavily dependent on data. Majority of this data however, is unstructured text coming from sources like emails, chats, social media, surveys, articles, and documents. The micro-blogging content coming from Twitter and Facebook poses serious challenges, not only because of the amount of data involved, but also because of the kind of language used in them to express sentiments, i.e., short forms, memes and emoticons.

Sentiment analysis also is used to monitor and analyse social phenomena, for the spotting of potentially dangerous situations and determining the general mood of the blogosphere.

Sentiment Analysis is also useful for practitioners and researchers, especially in fields like sociology, marketing, advertising, psychology, economics, and political science, which rely a lot on human-computer interaction data.

Sentiment Analysis enables companies to make sense out of data by being able to automate this entire process! Thus they are able to elicit vital insights from a vast unstructured dataset without having to manually indulge with it.

VADER Sentiment Analysis[edit]

VADER stands for Valence Aware Dictionary and Sentiment Reasoner.It is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. VADER uses a combination of A sentiment lexicon is a list of lexical features (e.g., words) which are generally labelled according to their semantic orientation as either positive or negative.

VADER has been found to be quite successful when dealing with social media texts. This is because VADER not only tells about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is.

It is fully open-sourced under the MIT License.

Code For Vader Sentiment Analysis[edit]

Posted By: divya98

```
# import SentimentIntensityAnalyzer class
# from vaderSentiment.vaderSentiment module.
from vaderSentiment.vaderSentiment import
SentimentIntensityAnalyzer

# function to print sentiments
# of the sentence.
def sentiment_scores(sentence):

    # Create a SentimentIntensityAnalyzer object.
    sid_obj = SentimentIntensityAnalyzer()
```

```
11
12      # polarity_scores method of SentimentIntensityAnalyzer
13      # oject gives a sentiment dictionary.
14      # which contains pos, neg, neu, and compound scores.
15      sentiment_dict = sid_obj.polarity_scores(sentence)
16
17      print("Overall sentiment dictionary is : ", sentiment_dict)
18      print("sentence was rated as ", sentiment_dict['neg']*100,
"% Negative")
19      print("sentence was rated as ", sentiment_dict['neu']*100,
"% Neutral")
20      print("sentence was rated as ", sentiment_dict['pos']*100,
"% Positive")
21
22      print("Sentence Overall Rated As", end = " ")
23
24      # decide sentiment as positive, negative and neutral
25      if sentiment_dict['compound'] >= 0.05 :
26          print("Positive")
27
28      elif sentiment_dict['compound'] <= - 0.05 :
29          print("Negative")
30
31      else :
32          print("Neutral")
33
34
35
36 # Driver code
37 if __name__ == "__main__" :
38
39      print("\n1st statement :")
40      sentence = "Joy of computing using python is the best course
for
41                  the computer science engineering students."
42
```

```
    # function calling
    sentiment_scores(sentence)

    print("\n2nd Statement :")
    sentence = "study is going on as usual"
    sentiment_scores(sentence)

    print("\n3rd Statement :")
    sentence = "I am vey sad today."
    sentiment_scores(sentence)
```

Advantages of Using VADER[edit]

VADER has a lot of advantages over traditional methods of Sentiment Analysis, including:

- It works exceedingly well on social media type text, yet readily generalizes to multiple domains
- It doesn't require any training data but is constructed from a generalizable, valence-based, human-curated gold standard sentiment lexicon
- It is fast enough to be used online with streaming data, and
- It does not severely suffer from a speed-performance tradeoff.

PANDAS[edit]

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word **Pan**el **Da**ta – an Econometrics from Multidimensional data.

Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Key Features of PANDAS[edit]

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and sub-setting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

Adaptive Histogram Equalization[edit]

Adaptive histogram equalization (AHE) is a computer image processing technique used to improve contrast in images. It differs from ordinary histogram equalization in the respect that the adaptive method computes several histograms, each corresponding to a distinct section of the image, and uses them to redistribute the lightness values of the image. It is therefore suitable for improving the local contrast and enhancing the definitions of edges in each region of an image.

However, AHE has a tendency to overamplify noise in relatively homogeneous regions of an image. A variant of adaptive histogram equalization called **Contrast Limited Adaptive Histogram Equalization** (**CLAHE**) prevents this by limiting the amplification.

Properties of AHE

- The size of the neighbourhood region is a parameter of the method. It constitutes a characteristic length scale: contrast at smaller scales is enhanced, while contrast at larger scales is reduced.
- Due to the nature of histogram equalization, the result value of a pixel under AHE is proportional to its rank among the pixels in its neighbourhood. This allows an efficient

implementation on specialist hardware that can compare the center pixel with all other pixels in the neighbourhood.

- An unnormalized result value can be computed by adding 2 for each pixel with a smaller value than the center pixel, and adding 1 for each pixel with equal value.
- When the image region containing a pixel's neighbourhood is fairly homogeneous regarding to intensities, its histogram will be strongly peaked, and the transformation function will map a narrow range of pixel values to the whole range of the result image. This causes AHE to overamplify small amounts of noise in largely homogeneous regions of the image.

What is OpenCV ?[edit]

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. In simple language it is library used for Image Processing. It is mainly used to do all the operation related to Images.

What it can do :

- Read and Write Images.
- Detection of faces and its features.
- Detection of shapes like Circle,rectangle etc in a image. E.g Detection of coin in images.
- Text recognition in images. e.g Reading Number Plates/
- Modifying image quality and colors e.g Instagram, CamScanner.
- Developing Augmented reality apps.
- and many more.....

Some Advantages of using OpenCV :

- Simple to learn,lots of tutorial available.
- Works with almost all the famous languages.
- Free to use.

==**Another solution to Upper triangular matrix problem**==

Lecture Notes:Week 9

From JOCWiki

Contents

[hide]

- 1 **Some Basic Programs**
  - 1.1 *GCD of two given numbers!*
    - 1.1.1 *1st LOGIC*
    - 1.1.2 *2nd LOGIC*
    - 1.1.3 *3rd LOGIC*
  - 1.2 *LCM of two given numbers!*
    - 1.2.1 *1st Logic*
    - 1.2.2 *2nd Logic*
  - 1.3 *Checking whether a given number is prime or not.*
  - 1.4 *To check whether a given string is Pallindrome or not...*
  - 1.5 *To print multiplication table of a given number*
  - 1.6 *Program to print prime series.*
  - 1.7 *To calculate the factorial of a number*
- 2 **Pattern printing (Alphabet Rangoli)**
- 3 **Another Logic for ANAGRAMS**
- 4 **Mathematical Functions in Python**
  - 4.1 *math.ceil(x)*
  - 4.2 *math.floor(x)*
  - 4.3 *math.factorial(x)*
  - 4.4 *math.fabs(x)*
  - 4.5 *math.exp(x)*
  - 4.6 *math.log(x,base)*
  - 4.7 *math.sqrt(x)*
  - 4.8 *math.modf(x)*
- 5 **Some Programs Revisited**
  - 5.1 Removing Duplicate Elements
  - 5.2 Panagrams
  - 5.3 *Removing Consecutive Vowels*
- 6 Turtle Methods
  - 6.1 Turtle Motion
    - 6.1.1 Movement
    - 6.1.2 Position
  - 6.2 Pen
    - 6.2.1 Draw
    - 6.2.2 Colour
    - 6.2.3 Appearance
- 7 **Program to Find the Resolution of JPEG Image**
- 8 **Natural Language Processing & NLTK**
  - 8.1 *Stylometry*
    - 8.1.1 **Author Stylometry**
  - 8.2 *Sentiment Analysis of a given sentence*
  - 8.3 *Tokenization*
  - 8.4 Additional Resource for Python Stylometry and **download links**
- 9 Matrix Equality
- 10 *Program to remove duplicate elements*
- 11 *Panagram*
- 12 *Program to remove consecutive vowels*
- 13 *Transpose of a Matrix*
- 14 *Python program to Transpose a Matrix*

- 15 *Generate a graph using Dictionary*
- 16 *Python Function to generate graph:*
- 17 *To generate the path from one node to the other node*
- 18 *Program to generate all the possible paths from one node to the other.*
- 19 *Program to generate the shortest path.*
- 20 *Printing ASCII values*
- 21 **NETWORKX**
- 22 **Why NetworkX ?**
- 23 **Graph**
- 24 **Undirected and Directed graph**
- 25 **Connected and Disconnected Graph:**
- 26 **Different types of graph generators**
- 27 Object Oriented Programming & Python
- 28 __init__ Method
- 29 Methods
- 30 Object Lifecycle
- 31 Inheritance
- 32 **Different Drawings Using NetworkX**
  - 32.1 **Simple Path**
  - 32.2 **Node Colormap**
  - 32.3 **Edge Colormap**
  - 32.4 **House With Colors**
  - 32.5 **Circular Tree**
  - 32.6 **Degree Rank**
  - 32.7 **Ego Graph**
  - 32.8 **Four Grids**
  - 32.9 **Degree Histogram**
  - 32.10 **Random Geometric Graph**
  - 32.11 **Weighted Graph**
  - 32.12 **Directed Graph**
  - 32.13 **Lanl Routes**
- 33 **The Erdős-Rényi Random Graph**
- 34 **Named Entity Recognition With NLTK**

Some Basic Programs[edit]

Here are some basic programs that one must definitely know!

- GCD of two given numbers.
- LCM of two given numbers.
- Checking whether a given number is prime or not.
- To check whether a given string is Pallindrome or not.
- To print multiplication table of a given number.

GCD of two given numbers![edit]

1st LOGIC[edit]

```
1 #Program for GCD of two numbers...
2 n,m = map(int,input('Enter two numbers ').split())
3 if n < m :
4     divisor = n
5     dividend = m
6 else:
```

```
    divisor = m
    dividend = n
rem = dividend%divisor
while rem != 0:
    dividend = divisor
    divisor = rem
    rem = dividend%divisor
print(f'GCD of {n} and {m} is {divisor}.')
'''
OUTPUT:
Enter two numbers 42 36
GCD of 42 and 36 is 6.
'''
```

2nd LOGIC[edit]

```
def GCD(x,y):
    import math
    diff = int(math.fabs(x-y))
    minimum = min(x,y)
    while diff!=x and diff!=y:
        x = diff
        y = minimum
        diff = int(math.fabs(x-y))
        minimum = min(x,y)
    return diff
'''
INPUT:
m,n = 16,8
a = GCD(m,n)
print(f'HCF of {m} and {n} is {a}')
OUTPUT:
HCF of 16 and 8 is 8
'''
```

3rd LOGIC[edit]

```
#GCD Using Recursion...
def GCD(num1,num2):
    if num1 < num2 :
        divisor = num1
        dividend = num2
    else:
        divisor = num2
        dividend = num1
    rem = dividend%divisor
    if rem==0:
        return divisor
    else:
        return GCD(divisor,rem)


'''
OUTPUT:
GCD(9,30)
3
'''
```

LCM of two given numbers![edit]

1st Logic[edit]

```
n,m = map(int,input('Enter two numbers ').split())
if n < m :
    divisor = n
    dividend = m
else:
    divisor = m
    dividend = n
rem = dividend%divisor
while rem != 0:
    dividend = divisor
    divisor = rem
    rem = dividend%divisor
lcm = int((m*n)/divisor)
print(f'LCM of {n} and {m} is {lcm}.')
```

```
'''
OUTPUT:
Enter two numbers 9 8
LCM of 9 and 8 is 72.
'''
```

2nd Logic[edit]

Checking whether a given number is prime or not.[edit]

```
def prime(n):
    if n == 0 or n == 1:
        return (f'{n} is not a prime number')
    else:
        for i in range(2,int(n/2)+1):
            if n%i == 0:
                return (f'{n} is not a prime number')
        return (f'{n} is a prime number')
'''
OUTPUT:
prime(4)
'4 is not a prime number'
'''
```

To check whether a given string is Pallindrome or not...[edit]

```
a = input()
b = ''
for i in range(len(a)-1,-1,-1):
    b = b + a[i]
if a == b:
    print(f'{a} is a PALLINDROME')
else:
    print(f'{a} is not a PALLINDROME')
'''
OUTPUT:
malayalam
```

```
malayalam is a PALLINDROME
'''
```

To print multiplication table of a given number[edit]

```
def multiplication(n):
    for i in range(1,11):
        print(f'{n} X {i} = {n*i}')

'''
OUTPUT:
multiplication(8)
8 X 1 = 8
8 X 2 = 16
8 X 3 = 24
8 X 4 = 32
8 X 5 = 40
8 X 6 = 48
8 X 7 = 56
8 X 8 = 64
8 X 9 = 72
8 X 10 = 80
'''
```

Program to print prime series.[edit]

```
start = input()
end = input()

for val in range(start, end + 1):

   if val > 1:
       for n in range(2, val):
           if (val % n) == 0:
               break
```

```
10        else:
11            print(val)
```

To calculate the factorial of a number[edit]

```
1 def fact(n):
2     if n == 0 or n == 1:
3         return 1
4     elif n < 0:
5         return 'Factorial is defined only for positive numbers'
6     else:
7         return n*fact(n-1)
'''
INPUT:
for i in range(-2,8):
    print(fact(i))
OUTPUT:
Factorial is defined only for positive numbers
Factorial is defined only for positive numbers
1
1
2
6
24
120
720
5040
'''
```

Pattern printing (Alphabet Rangoli)[edit]

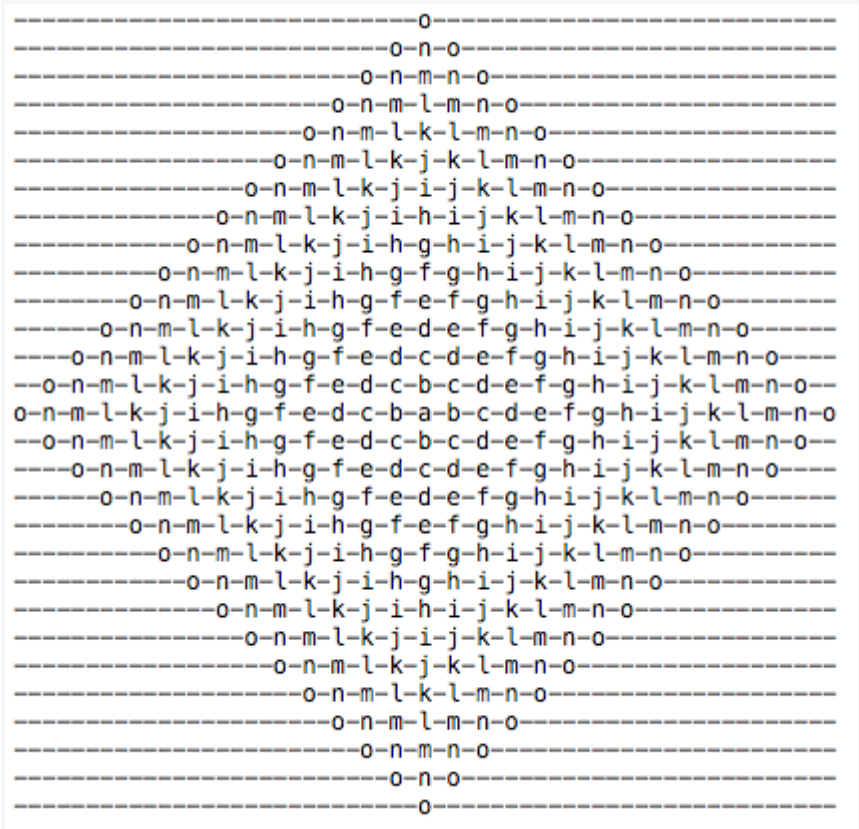
```
def print_rangoli(size):
    english = 'abcdefghijklmnopqrstuvwxyz'
    rows = size*2 - 1 #9
    columns = rows + (rows-1) #17
```

```
a = []
for i in range(rows):
    l = []
    for j in range(columns):
        l.append('-')
    a.append(l)
k = rows-1 #9
j = 1
for i in range(size):#Correct this
    m = k ; x = j ; q = size
    while x > 0:
        a[i][m] = english[q-1]
        m += 2
        x -= 1
        if x <= int(j/2):
            q += 1
        else:
            q -= 1
    j+=2
    k-=2


k = 2
j = rows - 2 #7
for i in range(size,rows):#Correct this
    m = k ; x = j ; q = size
    while x > 0:
        a[i][m] = english[q-1]
        m += 2
        x -= 1
        if x <= int(j/2):
            q += 1
        else:
            q -= 1
    j-=2
    k+=2
```

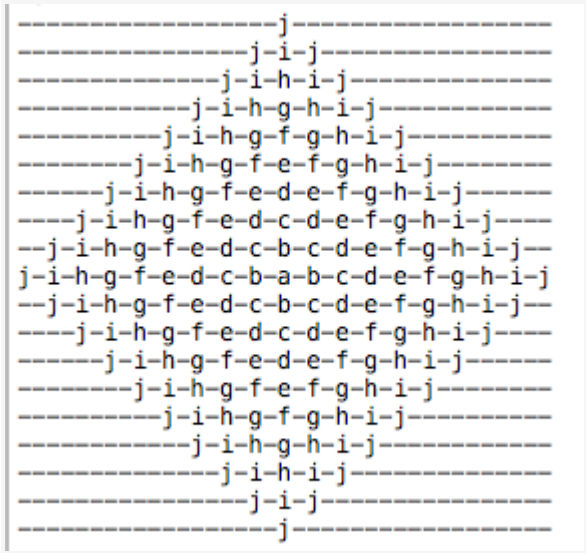
```
    for i in range(rows):
        for j in range(columns):
            print(a[i][j],end='')
        print()


if __name__ == '__main__':
    n = int(input())
    print_rangoli(n)
```

OUTPUT

For n=15



For n=10



Another Logic for ANAGRAMS [edit]

```
 1  #ANAGRAMS New Logic
 2 a = input("Enter the first string ").lower()
 3 b = input("Enter the second string ").lower()
 4 d = {};flag = True
 5 for i in a:
 6     if i not in d:
 7         d[i] = a.count(i)
 8 for i in b:
 9     if i not in d:
10         print("NOT ANAGRAM")
11         flag = False
12         break
13     elif d[i] == b.count(i):
14         pass
15     else:
16         print("NOT ANAGRAM")
17         flag = False
18         break
19 if flag == True: #If it has passed the entire 'for' loop without
breaking, then flag is still True
20     print("ANAGRAM")
```

```
'''
OUTPUT:
Enter the first string lists
Enter the second string slits
ANAGRAM
'''
```

Mathematical Functions in Python[edit]

- There are various mathematical functions in python that comes in handy while programming
- Some of them are listed below

math.ceil(x)[edit]

- Returns the smallest integer value greater than or equal to x.
- Note: It returns an integer value.

```
1 import math
2 for i in [1.0,1.2,1.4,1.6,1.8,2.0,2.2]:
3     print(f'ceil of {i} is: {math.ceil(i)}')
'''
OUTPUT:
ceil of 1.0 is: 1
ceil of 1.2 is: 2
ceil of 1.4 is: 2
ceil of 1.6 is: 2
ceil of 1.8 is: 2
ceil of 2.0 is: 2
ceil of 2.2 is: 3
'''
```

math.floor(x)[edit]

- Returns the largest integer value less than or equal to x.
- Note: It returns an integer value.

```
1
2 import math
3 for i in [1.0,1.2,1.4,1.6,1.8,2.0,2.2]:
4     print(f'floor of {i} is: {math.floor(i)}')
```

```
'''
OUTPUT:
floor of 1.0 is: 1
floor of 1.2 is: 1
floor of 1.4 is: 1
floor of 1.6 is: 1
floor of 1.8 is: 1
floor of 2.0 is: 2
floor of 2.2 is: 2
'''
```

math.factorial(x)[edit]

- Returns factorial of x.
- Raises ValueError if x is not integral or is negative.

```
for i in range(10):
    print(f'Factorial of {i} is {math.factorial(i)}')
```

```
'''
OUTPUT:
Factorial of 0 is 1
Factorial of 1 is 1
Factorial of 2 is 2
Factorial of 3 is 6
Factorial of 4 is 24
Factorial of 5 is 120
Factorial of 6 is 720
Factorial of 7 is 5040
Factorial of 8 is 40320
Factorial of 9 is 362880
'''
```

math.fabs(x) [edit]

- Return the absolute value of x.
- Note this returns a float value.

```
a = math.fabs(10-8)
b = math.fabs(8-10)
```

```
print('a is:',a)
print('b is:',b)

'''
OUTPUT:
a is: 2.0
b is: 2.0
'''
```

math.exp(x)[edit]

- Return e**x.
- Returns in float.

```
for i in range(10):
    print(math.exp(i))
'''
OUTPUT:
1.0
2.718281828459045
7.38905609893065
20.085536923187668
54.598150033144236
148.4131591025766
403.4287934927351
1096.6331584284585
2980.9579870417283
8103.083927575384
'''
```

math.log(x,base)[edit]

- With one argument, return the natural logarithm of x (to base e)
- With two arguments, return the logarithm of x to the given base, calculated as log(x)/log(base).
- Returns a float value.

```
 math.log(10)
#OUTPUT: 2.302585092994046
math.log(8,2)
```

```
4 #OUTPUT: 3.0
5 math.log(16,4)
6 #OUTPUT: 2.0
7 math.log(1)
8 #OUTPUT: 0.0
```

math.sqrt(x)[edit]

- Returns the square root of x.
- Note: Returns a float value.

```
1  for i in range(11):
2     print(f'Square Root of {i**2} is {math.sqrt(i**2)}')
'''
OUTPUT:
Square Root of 0 is 0.0
Square Root of 1 is 1.0
Square Root of 4 is 2.0
Square Root of 9 is 3.0
Square Root of 16 is 4.0
Square Root of 25 is 5.0
Square Root of 36 is 6.0
Square Root of 49 is 7.0
Square Root of 64 is 8.0
Square Root of 81 is 9.0
Square Root of 100 is 10.0
'''
```

NOTE: Value Error, if negative number is provided.

math.modf(x)[edit]

- Returns the fractional and integer parts of x.
- Both results carry the sign of x and are floats.

```
1 import math
2 a = math.modf(2.64)
3 type(a)
4 #OUTPUT: tuple
5 print(a)
6 #OUTPUT: (0.6400000000000001, 2.0)
```

Some Programs Revisited[edit]

Removing Duplicate Elements[edit]

```
 1  l = list(map(int,input().split()))
 2 m = []
 3 for i in l:
 4   if i in m:
 5     pass
 6   else:
 7     m.append(i)
 8 for i in range(len(m)):
 9   if i == len(m)-1:
10     print(m[i],end='')
11   else:
12     print(m[i],end=' ')
```

Panagrams[edit]

```
 1 import string
 2 a = input().lower()
 3 b = list(string.ascii_lowercase)
 4 for i in b:
 5   if i in a:
 6     flag = 0
 7   else:
 8     flag = 1
 9     break
10 if flag == 0:
11   print("YES",end='')
12 else:
13   print("NO",end='')
```

Removing Consecutive Vowels[edit]

```
 1  b = input().lower()
 2 a = []
 3 for i in b:
```

```
    a.append(i)

vowel = ['a','e','i','o','u']
for i in range(len(b)-1):
    if b[i] in vowel and b[i+1] in vowel:
        a[i+1] = 0
for i in a:
        if i == 0:
            pass
        else:
            print(i,end='')


'''
INPUT:
three four five
OUTPUT:
thre for five

'''
```

Turtle Methods[edit]

- **Turtle Motion[edit]**
  1. ***Movement[edit]***

forward( )
backward( )
left( )
right( )
goto( )
setpos( )
setposition( )
setx( )
sety( )
setheading( )
seth( )

  2. ***Position[edit]***

position( )
towards( )
heading( )

distance( )
xcor( )
ycor( )
degrees( )
radians( )

- **Pen**[edit]
  1. ***Draw***[edit]

pensize( )
width( )
pen( )
isdown( )
pendown( )
pd( )
down( ) penup( )
pu( )
up( )

  2. ***Colour***[edit]

fillcolor()
color()
pencolor()
fill()
begin_fill()
end_fill()

  3. ***Appearance***[edit]

reset()
clear()
write()

Program to Find the Resolution of JPEG Image[edit]

```
def jpeg_res(filename):
   """This function prints the resolution of the jpeg image file
passed into it"""

   # open image for reading in binary mode
   with open(filename,'rb') as img_file:

       # height of image (in 2 bytes) is at 164th position
       img_file.seek(163)
```

```
        # read the 2 bytes
        a = img_file.read(2)

        # calculate height
        height = (a[0] << 8) + a[1]

        # next 2 bytes is width
        a = img_file.read(2)

        # calculate width
        width = (a[0] << 8) + a[1]

    print("The resolution of the image is",width,"x",height)

jpeg_res("img1.jpg")
```

Natural Language Processing & NLTK[edit]

Stylometry[edit]

Stylometry is the quantitative study of literary style through computational distant reading methods. It is based on the observation that authors tend to write in relatively consistent, recognizable and unique ways. For example:

- Each person has their own unique vocabulary, sometimes rich, sometimes limited. Although a larger vocabulary is usually associated with literary quality, this is not always the case.
- Some people write in short sentences, while others prefer long blocks of text consisting of many clauses.
- No two people use semicolons, em-dashes, and other forms of punctuation in the exact same way.

Scholars have used stylometry as a tool to study a variety of cultural questions.

- For example, a considerable amount of research has studied the differences between the ways in which men and women write or are written about.
- Other scholars have studied the ways in which a sudden change in writing style within a single text may indicate plagiarism.

Author Stylometry[edit]

Author stylometry is the analysis of writing styles of various authors. It is based on the observation that various authors have unique writing styles. They use words and sentences of different lengths or same words often. So it is possible to identify the author of an anonymous text.

Sentiment Analysis of a given sentence[edit]

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.downloader.download('vader_lexicon')
sid = SentimentIntensityAnalyzer()
data = input("Enter the string ")
ss = sid.polarity_scores(data)
for k in ss:
    print(k,ss[k])
```

```
'''
OUTPUT:
Enter the string: SCCI labs of IIT Ropar deals with research areas
related to Social Computing.
neg 0.0
neu 1.0
pos 0.0
compound 0.0
'''
```

Tokenization[edit]

In Python tokenization basically refers to splitting up a larger body of text into smaller lines, words or even creating words for a non-English language. The various tokenization functions in-built into the nltk module itself and can be used.

Line Tokenization

In the example we divide a given text into different lines by using the function sent_tokenize.

```
import nltk
sentence_data = "The First sentence is about Python. The Second:
about Django. You can learn Python, Django and Data Ananlysis here."
nltk_tokens = nltk.sent_tokenize(sentence_data)
print (nltk_tokens)
```

```
'''
Output:
```

```
['The First sentence is about Python.', 'The Second: about Django.',
'You can learn Python, Django and Data Ananlysis here.']
'''
```

Word Tokenzitaion
We tokenize the words using word_tokenize function available as part of nltk.

```
1 import nltk
2 word_data = "It originated from the idea that there are readers who
prefer learning new skills from the comforts of their drawing rooms"
3 nltk_tokens = nltk.word_tokenize(word_data)
4 print (nltk_tokens)
```

```
'''
Output:
['It', 'originated', 'from', 'the', 'idea', 'that', 'there', 'are',
'readers',
'who', 'prefer', 'learning', 'new', 'skills', 'from', 'the',
'comforts', 'of', 'their', 'drawing', 'rooms']
'''
```

Additional Resource for Python Stylometry and download links[edit]

Introduction to stylometry with Python:
Below website has lessons which will help you learn to conduct 'stolymetric analysis' on texts and determine authorship of disputed texts. Those lessons cover three methods: Mendenhall's Characteristic Curves of Composition, Kilgariff's Chi-Squared Method, and John Burrows' Delta Method.

https://programminghistorian.org/en/lessons/introduction-to-stylometry-with-python

Link to download **stylometry-federalist.zip**

Matrix Equality[edit]

Two matrices are said to be equal if they satisfy all of the following criteria:

- Both matrix should have the same number of rows.
- Both matrix should have the same number of columns.

- All the corresponding elements within each matrix are equal.

Program to remove duplicate elements[edit]

```
l = [int(x) for x in input().split(" ")]
c=[]
for i in l:
    if i not in c:
      c.append(i)
for i in c:
  print(i,end=' ')
```

Panagram[edit]

```
sentence=input()
def checkPangram(s):
    List = []
    for i in range(26):
        List.append(False)
    for c in s.lower():
        if not c == " ":
            List[ord(c) -ord('a')]=True
    for ch in List:
        if ch == False:
            return False
    return True
if (checkPangram(sentence)):
    print("YES")
else:
    print("NO")
```

Program to remove consecutive vowels[edit]

```
str=input()
def is_vow(c):
    return ((c == 'a') or (c == 'e') or
```

```
            (c == 'i') or (c == 'o') or
            (c == 'u'));
def removeVowels(str):

    print(str[0], end = "");
    for i in range(1,len(str)):
        if ((is_vow(str[i - 1]) != True) or
            (is_vow(str[i]) != True)):

            print(str[i], end = "");

removeVowels(str);
```

Transpose of a Matrix[edit]

The transpose of a matrix is a new matrix whose rows are the columns of the original.
In which 1st row becomes the 1st column of a new matrix ,and 2nd row becomes the 2nd column of a matrix and do on.
The 'm*n' matrix after transpose becomes 'n*m' matrix.
The superscript "T" means "transpose".

Python program to Transpose a Matrix[edit]

```
original = []
inner = []
R = int(input("Enter the number of rows :"))
C = int(input("Enter the number of columns :"))

for i in range(0,R,1):
    inner = []
    for i in range(0,C,1):
        value = int(input())
        inner.append(value)
    original.append(inner)

print(original)

```

```
15
16 def Transpose(array,r,c):
17     transpose = []
18     for i in range(c):
19         inner = []
20         for j in range(r):
21             inner.append(array[j][i])
22         transpose.append(inner)
23     return transpose
24
25 print(Transpose(original,R,C))
```

Generate a graph using Dictionary [edit]

To implement graph using dictionary data structure in python.
The keys of the dictionary used are the nodes of our graph and the corresponding values are lists with each nodes, which are connecting by an edge.
This simple graph has six nodes (a-f) and five arcs:

```
a -> c
b -> c
b -> e
c -> a
c -> b
c -> d
c -> e
d -> c
e -> c
e -> b
```

This is a dictionary whose keys are the nodes of the graph. For each key, the corresponding value is a list containing the nodes that are connected by a direct arc from this node.

```
graph = { "a" : ["c"],
          "b" : ["c", "e"],
          "c" : ["a", "b", "d", "e"],
          "d" : ["c"],
          "e" : ["c", "b"],
          "f" : []
        }
```

Python Function to generate graph:[edit]

```
def generate_edges(graph):
    edges = []
    for node in graph:
        for neighbour in graph[node]:
            edges.append((node, neighbour))
    return edges
```

To generate the path from one node to the other node[edit]

Using Python dictionary, we can find the path from one node to the other in a Graph. The idea is similar to DFS in graphs. In the function, initially, the path is an empty list. In the starting, if the start node matches with the end node, the function will return the path. Otherwise the code goes forward and hits all the values of the starting node and searches for the path using recursion. If there is no such path, it returns None.

```
graph ={
'a':['c'],
'b':['d'],
'c':['e'],
'd':['a', 'd'],
'e':['b', 'c']
}

# function to find path
def find_path(graph, start, end, path =[]):
  path = path + [start]
  if start == end:
    return path
  for node in graph[start]:
    if node not in path:
      newpath = find_path(graph, node, end, path)
      if newpath:
        return newpath
      return None

# Driver function call to print the path
print(find_path(graph, 'd', 'c'))
```

1. Output:

```
['d', 'a', 'c']
```

Program to generate all the possible paths from one node to the other.[edit]

```
graph ={
'a':['c'],
'b':['d'],
'c':['e'],
'd':['a', 'd'],
'e':['b', 'c']
}
```

```
# function to generate all possible paths
def find_all_paths(graph, start, end, path =[]):
  path = path + [start]
  if start == end:
    return [path]
  paths = []
  for node in graph[start]:
    if node not in path:
      newpaths = find_all_paths(graph, node, end, path)
    for newpath in newpaths:
      paths.append(newpath)
  return paths


# Driver function call to print all
# generated paths
print(find_all_paths(graph, 'd', 'c'))
Output:

[['d', 'a', 'c'], ['d', 'a', 'c']]
```

Program to generate the shortest path.[edit]

To get to the shortest from all the paths, we use a little different approach as. In this, as we get the path from the start node to the end node, we compare the length of the path with a variable named as shortest which is initialized with the None value. If the length of generated path is less than the length of shortest, if shortest is not None, the newly generated path is set as the value of shortest. Again, if there is no path, it returns None.

```
graph ={
'a':['c'],
'b':['d'],
'c':['e'],
'd':['a', 'd'],
'e':['b', 'c']
}

# function to find the shortest path
def find_shortest_path(graph, start, end, path =[]):
        path = path + [start]
        if start == end:
            return path
        shortest = None
        for node in graph[start]:
            if node not in path:
                newpath = find_shortest_path(graph, node, end, path)
                if newpath:
                    if not shortest or len(newpath) < len(shortest):
                        shortest = newpath
        return shortest

# Driver function call to print
# the shortest path
print(find_shortest_path(graph, 'd', 'c'))
Output:

['d', 'a', 'c']
```

Printing ASCII values[edit]

```
a = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

```
2 for i in a:
3     print(f'ASCII Value for {i} is {ord(i)}')
'''
OUTPUT:
ASCII Value for a is 97
ASCII Value for b is 98
ASCII Value for c is 99
ASCII Value for d is 100
ASCII Value for e is 101
ASCII Value for f is 102
ASCII Value for g is 103
ASCII Value for h is 104
ASCII Value for i is 105
ASCII Value for j is 106
ASCII Value for k is 107
ASCII Value for l is 108
ASCII Value for m is 109
ASCII Value for n is 110
ASCII Value for o is 111
ASCII Value for p is 112
ASCII Value for q is 113
ASCII Value for r is 114
ASCII Value for s is 115
ASCII Value for t is 116
ASCII Value for u is 117
ASCII Value for v is 118
ASCII Value for w is 119
ASCII Value for x is 120
ASCII Value for y is 121
ASCII Value for z is 122
ASCII Value for A is 65
ASCII Value for B is 66
ASCII Value for C is 67
ASCII Value for D is 68
ASCII Value for E is 69
ASCII Value for F is 70
```

```
ASCII Value for G is 71
ASCII Value for H is 72
ASCII Value for I is 73
ASCII Value for J is 74
ASCII Value for K is 75
ASCII Value for L is 76
ASCII Value for M is 77
ASCII Value for N is 78
ASCII Value for O is 79
ASCII Value for P is 80
ASCII Value for Q is 81
ASCII Value for R is 82
ASCII Value for S is 83
ASCII Value for T is 84
ASCII Value for U is 85
ASCII Value for V is 86
ASCII Value for W is 87
ASCII Value for X is 88
ASCII Value for Y is 89
ASCII Value for Z is 90
```

NETWORKX[edit]

NetworkX in Python is used for creation, manipulation and study of structures of graphs and networks. NetworkX is used in mathematics, physics, biology, computer science, social networks, etc. We can also generate random networks, design new network algorithms and draw networks.

Command for installation: pip install networkx

(or) pip install networkx[all]

Upgrade networkx: pip install --upgrade networkx

- NetworkX is suitable for real-world graph problems and is good at handling big data as well.
- As the library is purely made in python, this fact makes it highly scalable, portable and reasonably efficient at the same time.
- It is open source and released under 3-clause BSD License.

Why NetworkX ?[edit]

NetworkX gives you a lot of reasons to go with it. Following are some features of NetworkX that makes it a package to go with:

- It has numerous standard graph algorithms
- It supports data structures for graphs, digraphs, and multigraphs
- It provides various network structure and measures for analysis
- Making classic/random graphs and synthetic networks is much easier using generators provided in the package
- Nodes in your network or graph can be absolutely anything, be it images, XML data or anything else
- Edges also can hold arbitrary data like timestamp and weight
- It has been well tested with about 90% code coverage

Graph[edit]

A graph is a represenatation of a network. It is collection of points, called nodes, and lines connecting those points, called edges. Nodes are also called vertices. We can unerstand a graph or a network with the help of an example. Let us say, there are five cities and there are different roads connecting them. So this network of cities and roads is a graph. The cities in this example are nodes and the roads between the cities are edges.

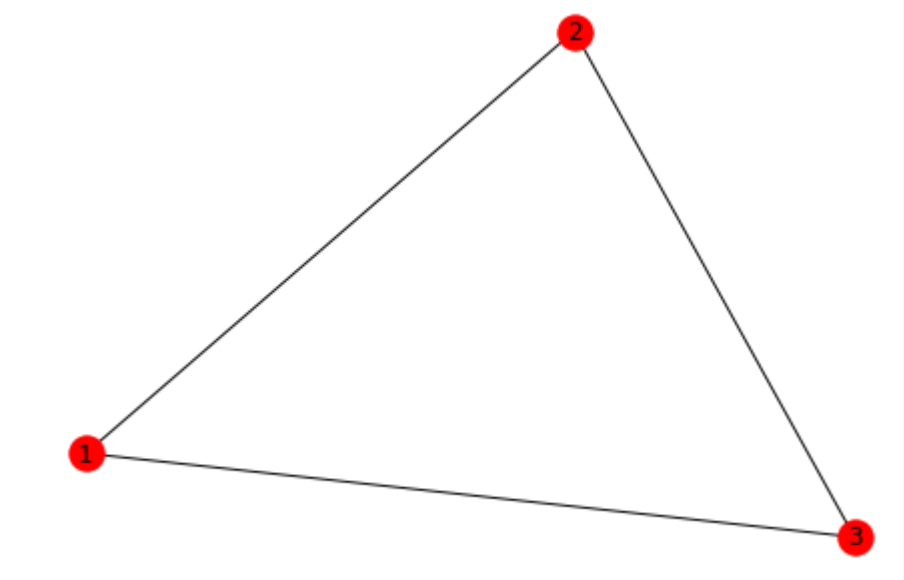
Other exmaples of graph:

```
a. Network of people on social media.
b. Network of webpages
c. Google maps
```

Creating a simple graph:

```
1  import networkx as nx
2  import matplotlib.pyplot as plt            #for plotting the graph
3
4  #creating a graph
5  G = nx.Graph()
6
7  #adding nodes 1,2 and 3
8  G.add_node(1)
9  G.add_node(2)
10 G.add_node(3)
11
12 #adding egdes connecting nodes
13 G.add_edge(1,2)
14 G.add_edge(2,3)
```

```
15  G.add_edge(1,3)
16
17  #drawing and displaying the graph
18  nx.draw(G, with_labels=True)   #label nodes also
19  plt.show()
```



Adding nodes from a list:

```
G = nx.Graph()

#list of nodes
l = [1,2,3,4,5]
#add nodes
G.add_nodes_from(l)
```

Getting nodes and edges

```
1  import networkx as nx
2  import matplotlib.pyplot as plt
3
4  G = nx.Graph()
5
6  #list of nodes
7  l = [1,2,3,4]
```

```
 8
 9  #add nodes
10  G.add_nodes_from(l)
11
12  #add edges
13  G.add_edge(1,2)
14  G.add_edge(2,3)
15  G.add_edge(2,4)
16
17  print('Nodes are ',G.nodes())
18  print('Edges are ',G.edges())
19
20  '''
21  Output:
22  Nodes are  [1, 2, 3, 4]
23  Edges are  [(1, 2), (2, 3), (2, 4)]
```

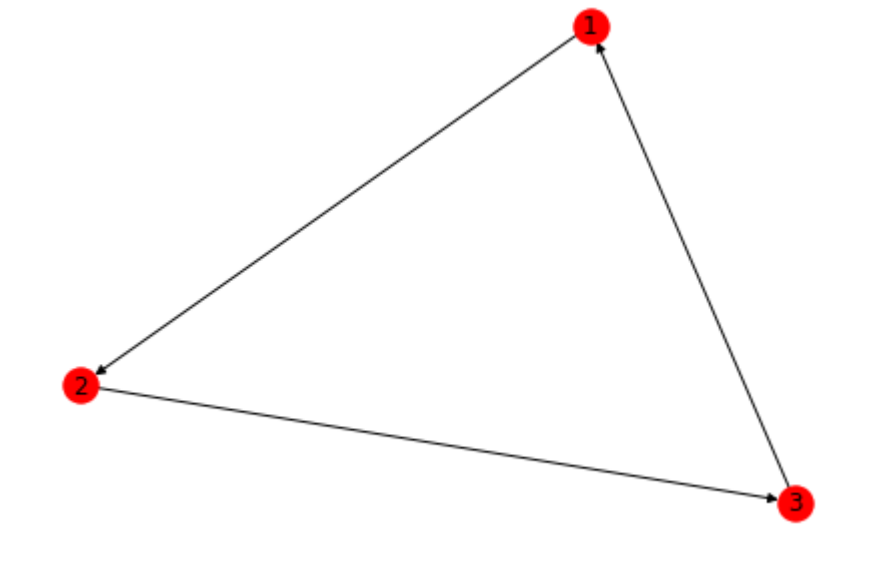
Degree of a Node:

```
The degree of a node is the number of edges from that node to other
nodes. It is basically the number of connections that
a node has with othe rnodes.
#get degree of a node
G.degree(1) #where 1 is node name
```

Undirected and Directed graph[edit]

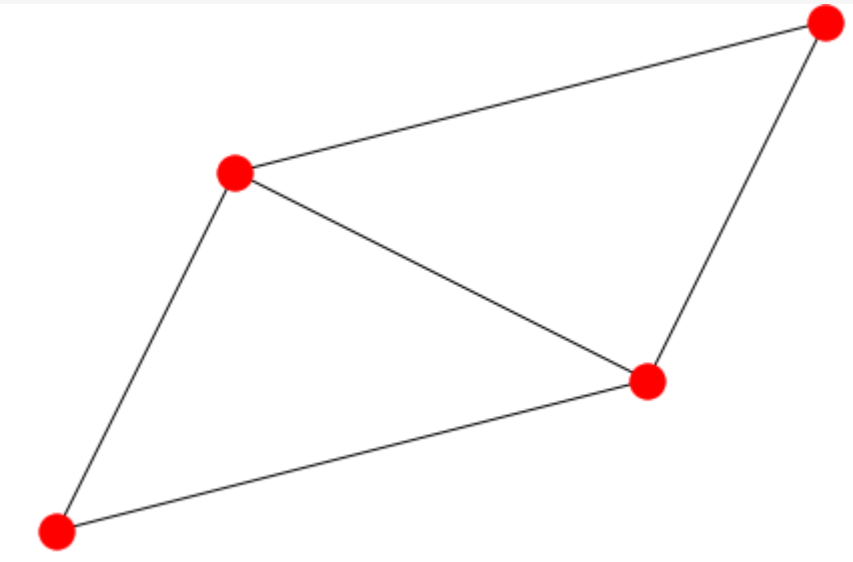
```
An undirected graph is a graph in which edges have no direction.
An undirected graph is created by G = nx.Graph()

A directed graph is a graph which has all the edges directed from one
node to another node. Every edge has a direction
associated with it. The directions are marked with an arrow.
Directed graph is created by DiGraph class such as G=nx.DiGraph()
```

The below is a directed graph:

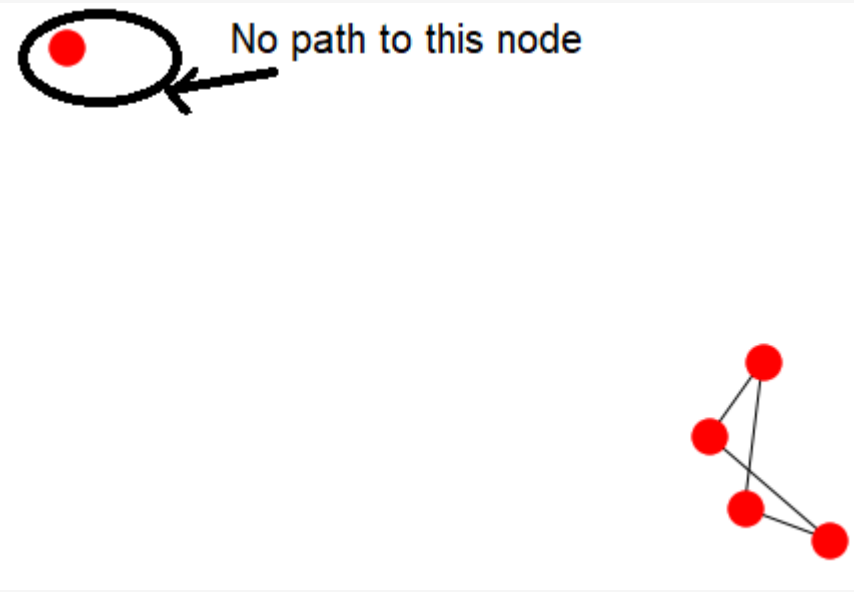


Connected and Disconnected Graph:[edit]

A connected graph is a graph in which every node is reachable i.e., there is a path to every node. It is a graph in which there is a path from one node to every other node. The path can be one or more than one edges long.
The below graph is connected.



The below graph is disconnected.



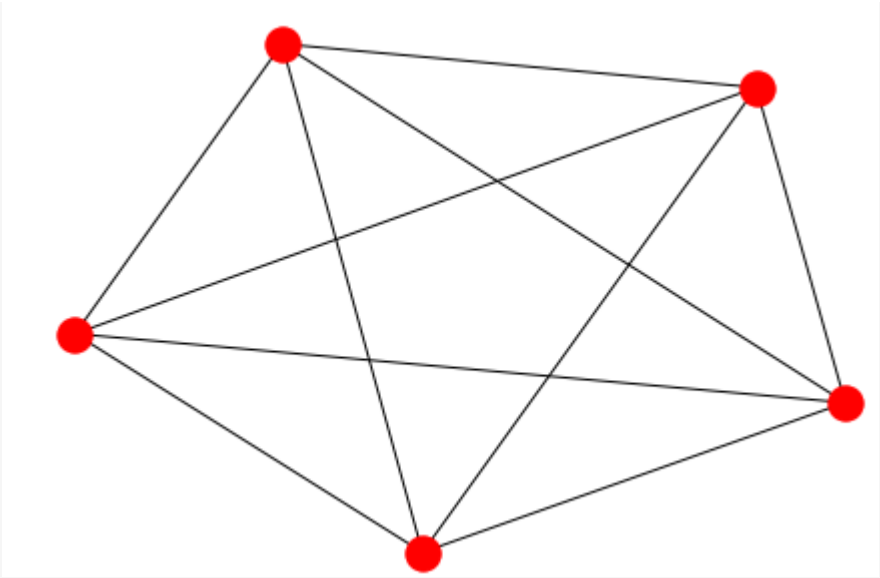
Different types of graph generators[edit]

1)Complete Graph:

A complete graph is a graph which has an edge from every single node to other node. In other words, there is an edge between every pair of nodes.

Example:

```
G = nx.complete_graph(5)
#This will create a graph of 5 nodes with every node connected.
```



2)Random Graph:

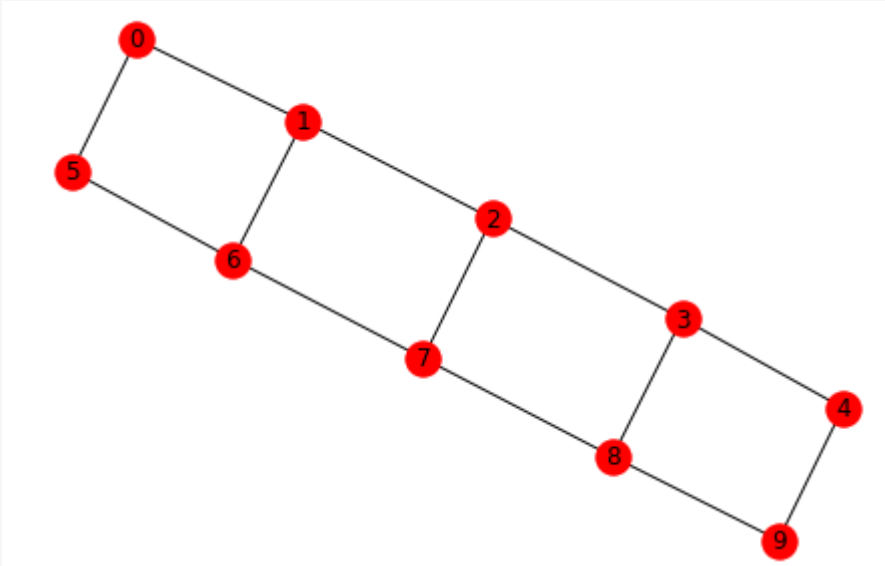
We can generate a random graph with gnp_random_graph() method.

```
G = nx.gnp_random_graph(n=20,p=0.5)
# Here n is the number of nodes and p is the probability of edge
creation between two nodes.
```

3)Ladder Graph:

We can generate a ladder shaped graph with ladder_graph() method.

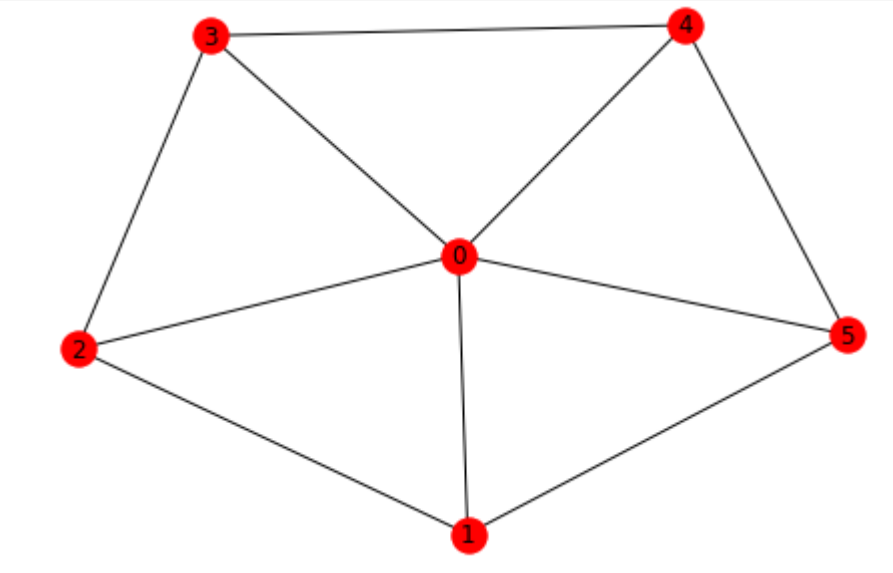
```
G = nx.ladder_graph(n=5)
# Here n is the number of nodes
```



4)Wheel Graph:

We can generate a ladder shaped graph with wheel_graph() method.

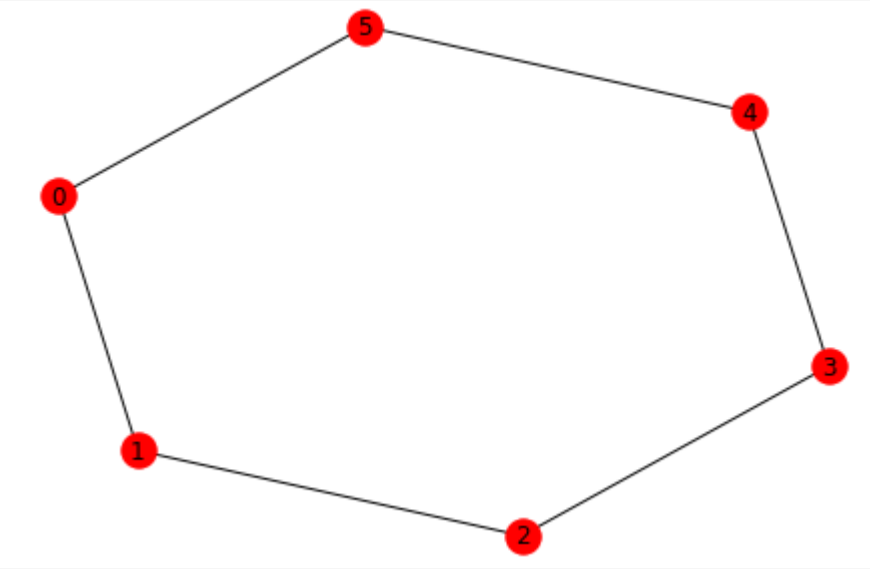
```
G = nx.wheel_graph(n=6)
# Here n is the number of nodes
```



5)Cycle Graph:

We can generate a cyclic graph with cycle_graph() method.

```
G = nx.cycle_graph(n=6)
# Here n is the number of nodes
```



5)Empty Graph:

We can generate a empty graph with empty_graph() method. This graph will not have any nodes.

```
G = nx.empty_graph(n=6)
# Here n is the number of nodes
```

Object Oriented Programming & Python[edit]

Object oriented programming (OOP) is an approach to program organisation & development that attempts to eliminate some of the pitfalls of conventional programming methods by incorporating the best of structured programming features with several powerful new concepts. It is a new way of organising & developing programs.

Objects are created using classes, which are actually the focal point of OOP.

The CLASS describes what the object will be, but is separate from the object itself. A class can be described as an object's blueprint, description or definition.

__init__ Method[edit]

The **__init__** method is the most important method in a class. This is called when an instance (object) of the class is created, using the class name as a function.

All methods MUST HAVE **self** as their first parameter, although it isn't explicitly passed, Python adds the self argument to the list for you; you do not need to include it when you call the methods.

Within a method definition, self refers to the instance calling the method.

In an **__init__** method, **self.attribute** can therefore be used to set the initial value of an instance's attribute.

The __init__ method is called the class constructor.

Example:

```
>>>class cat:
         def __init__(self, color, legs):
                     self.color = color
                     self.legs = legs


>>>felix = cat("ginger", 4)
>>>rover = cat("brown", 4)
```

Methods[edit]

Classes can have other methods defined to add functionality to them. Remember, all methods must have "self" as their first parameter.

Classes can also have class attributes, created by assigning variables within the body of the class. These can be accessed from either instances of the class or the class itself.

Class attributes are shared by all instances of the class. Trying to access an attribute of an instance that isn't defined causes an **AttributeError**. This also applies when we call an undefined method.

Object Lifecycle[edit]

- The lifecycle of an object is made up of its *creation*, *manipulation* & *destruction*.
- The first stage of the life cycle of an object is the definition of the class to which it belongs.
- The next stage is the instantiation of an instance, when *__init__* is called. Memory is allocated to store the instance.

Just before this occurs, the **__new__** method of the class is called. This is usually overridden only in special cases. After this, the object is ready to be used.

Eventually, it will finish being used & can be destroyed.

When an object is destroyed, the memory allocated to it is freed up & can be used for other purposes. Destruction of an object occurs when its **reference count** reaches zero.

{ **Reference Count** is the number of variables & other elements that refer to an object. }

If nothing is referring to it (it has a reference count = zero), nothing can interact with it, so, it can be safely deleted.

The **del** statement reduces the reference count of an object by one & this often leads to its deletion. The magic method for del statement is **__del__**.

The process of deleting objects when they are no longer needed is called **Garbage Collection**.

An object's reference count increases when it is assigned a new name or placed in a container (list, tuple, or dictionary).

The object's reference count decreases when it is deleted with **"del"**, its reference is **re-assigned**, or its reference **goes out of scope**.

When an object's reference count reaches ZERO, Python automatically deletes it (**Automatic Memory Management**).

Inheritance[edit]

Inheritance provides a way to share functionality between classes. To inherit a class from another class, put the superclass name in parenthesis after the class name.

- A class that **inherits from another class** is called **subclass**.
- A class that **is inherited from** is called a **superclass**.
- If a class inherits from another with the same attributes or methods, it **overrides** them.

```
class wolf:

     def __init__(self, name, color):

          self.name = name

          self.color = color
```

```
def bark(self):
    print(" ")


class dog(wolf):
      def bark(self):
          print(" ")


husky = dog("max", "grey")
husky.bark()
```

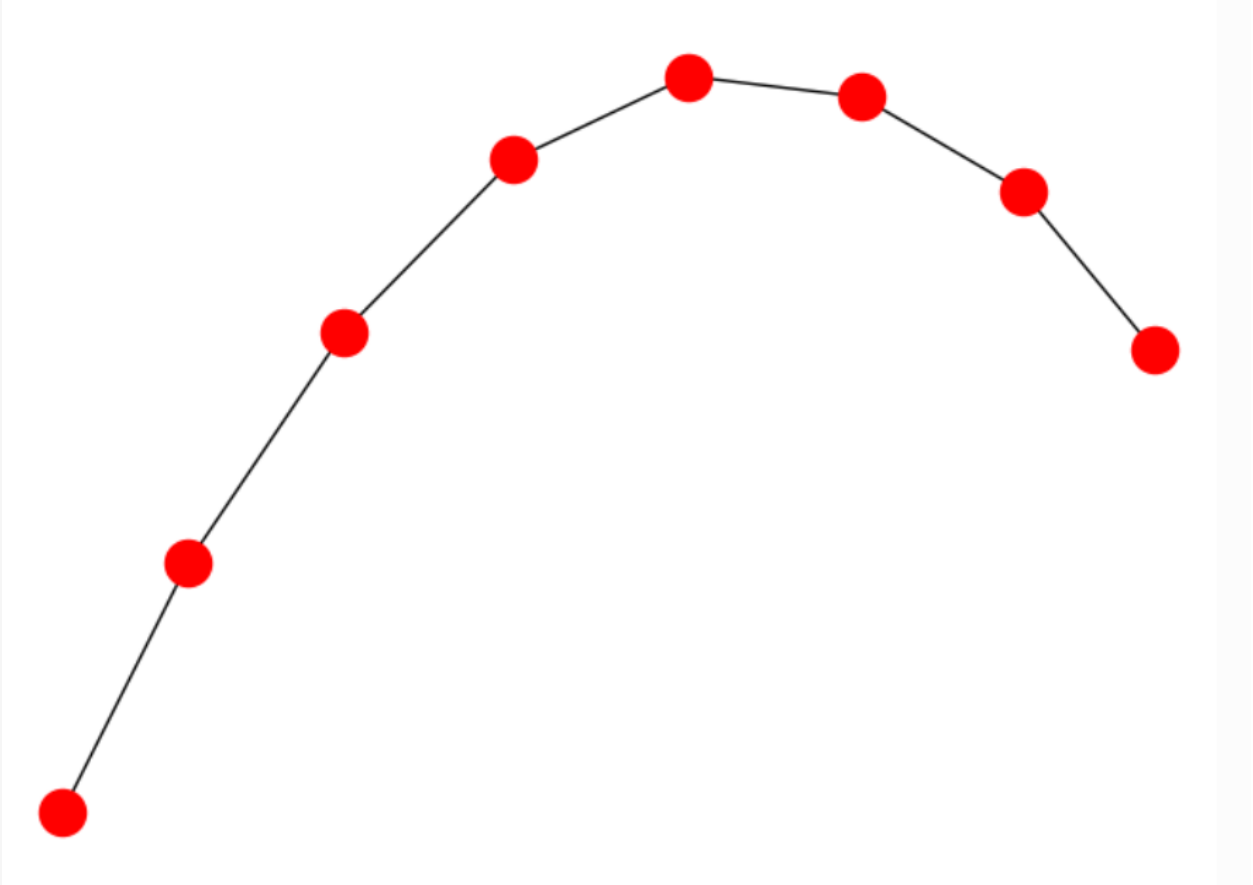
Inheritance can also be indirect. One class can inherit from another & that class can inherit from a third class. However, circular inheritance is NOT possible.

The function **super()** is a useful inheritance-related function that refers to the parent class. It can be used to find the method with a certain name in an object's superclass.

Different Drawings Using NetworkX[edit]

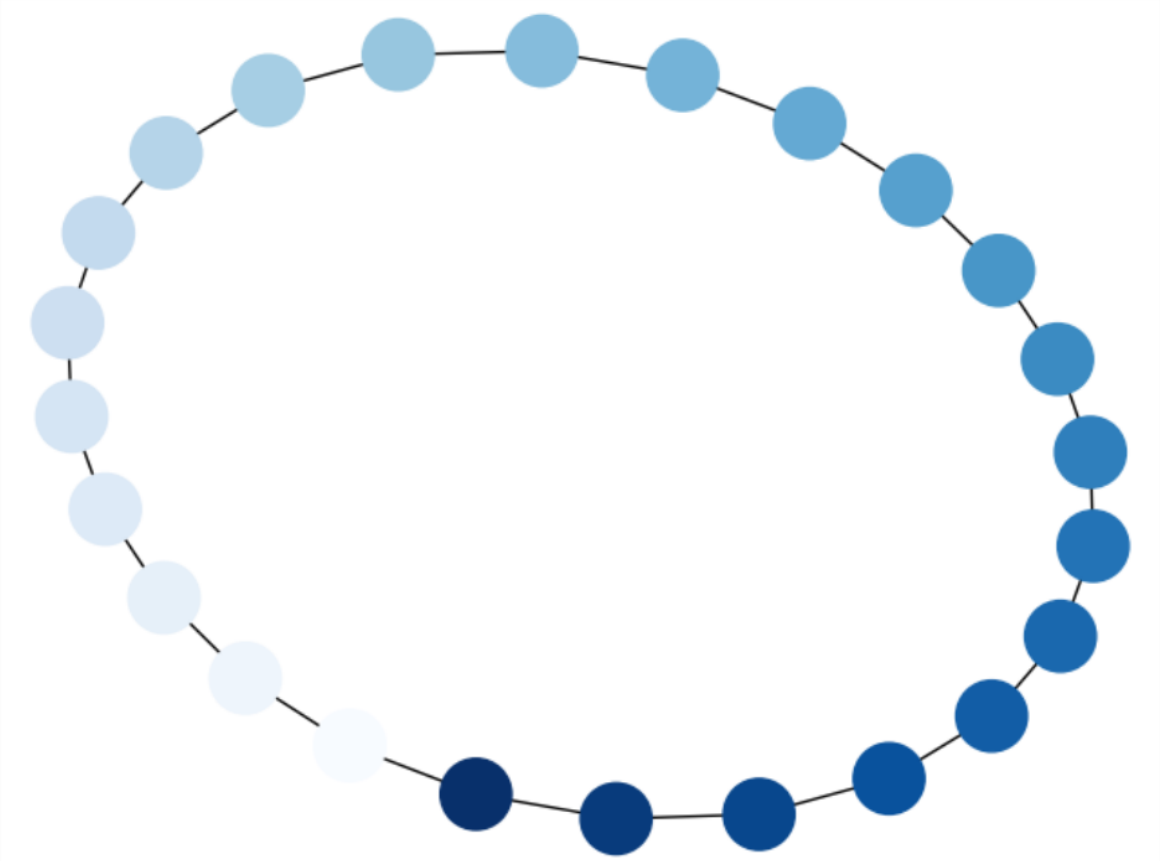
Simple Path[edit]

```
import matplotlib.pyplot as plt
import networkx as nx


G = nx.path_graph(8)
nx.draw(G)
plt.show()
```



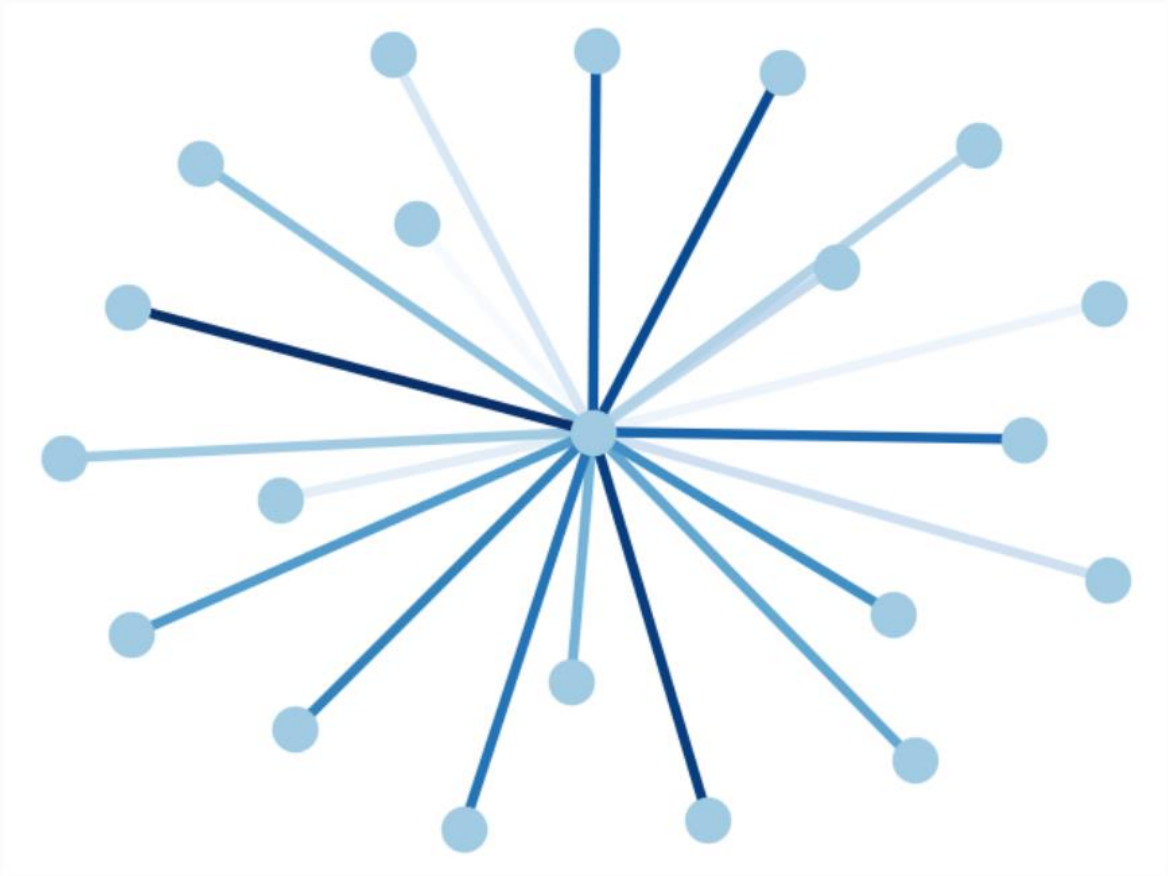
Node Colormap[edit]

```
import matplotlib.pyplot as plt
import networkx as nx

G = nx.cycle_graph(24)
pos = nx.spring_layout(G, iterations=200)
nx.draw(G, pos, node_color=range(24), node_size=800, cmap=plt.cm.Blues)
plt.show()
```



Edge Colormap[edit]

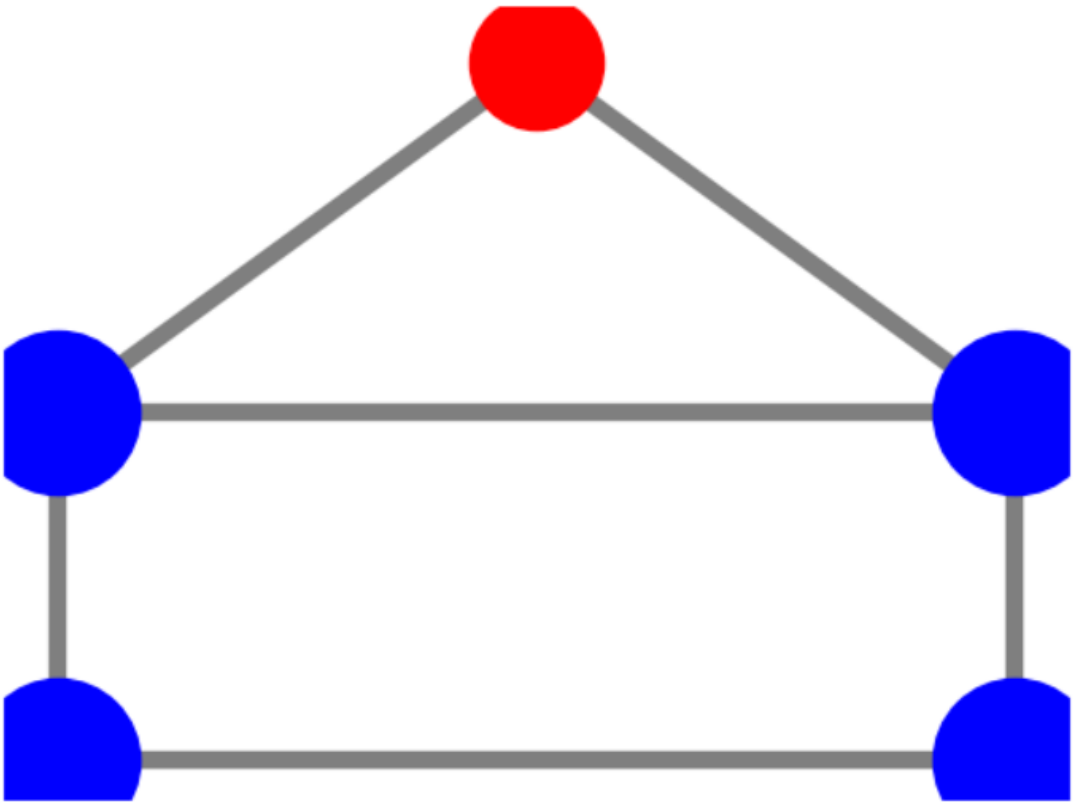
```
import matplotlib.pyplot as plt
import networkx as nx

G = nx.star_graph(20)
pos = nx.spring_layout(G)
colors = range(20)
nx.draw(G, pos, node_color='#A0CBE2', edge_color=colors,
        width=4, edge_cmap=plt.cm.Blues, with_labels=False)
plt.show()
```



House With Colors[edit]

```
import matplotlib.pyplot as plt
import networkx as nx

G = nx.house_graph()
# explicitly set positions
pos = {0: (0, 0),
       1: (1, 0),
       2: (0, 1),
       3: (1, 1),
       4: (0.5, 2.0)}

nx.draw_networkx_nodes(G, pos, node_size=2000, nodelist=[4])
nx.draw_networkx_nodes(G, pos, node_size=3000, nodelist=[0, 1, 2, 3],
node_color='b')
nx.draw_networkx_edges(G, pos, alpha=0.5, width=6)
plt.axis('off')
```

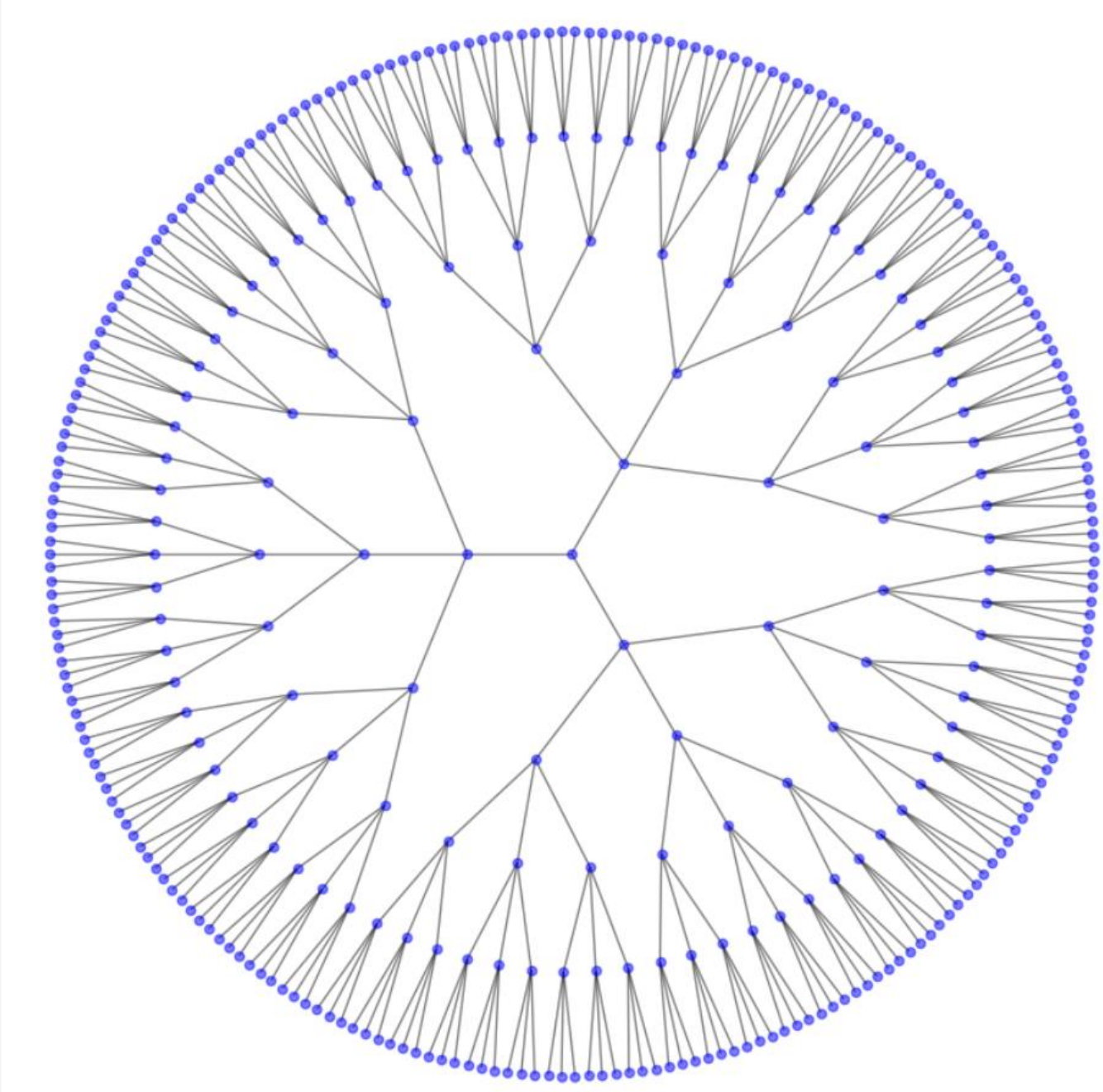
```
plt.show()
```



Circular Tree[edit]

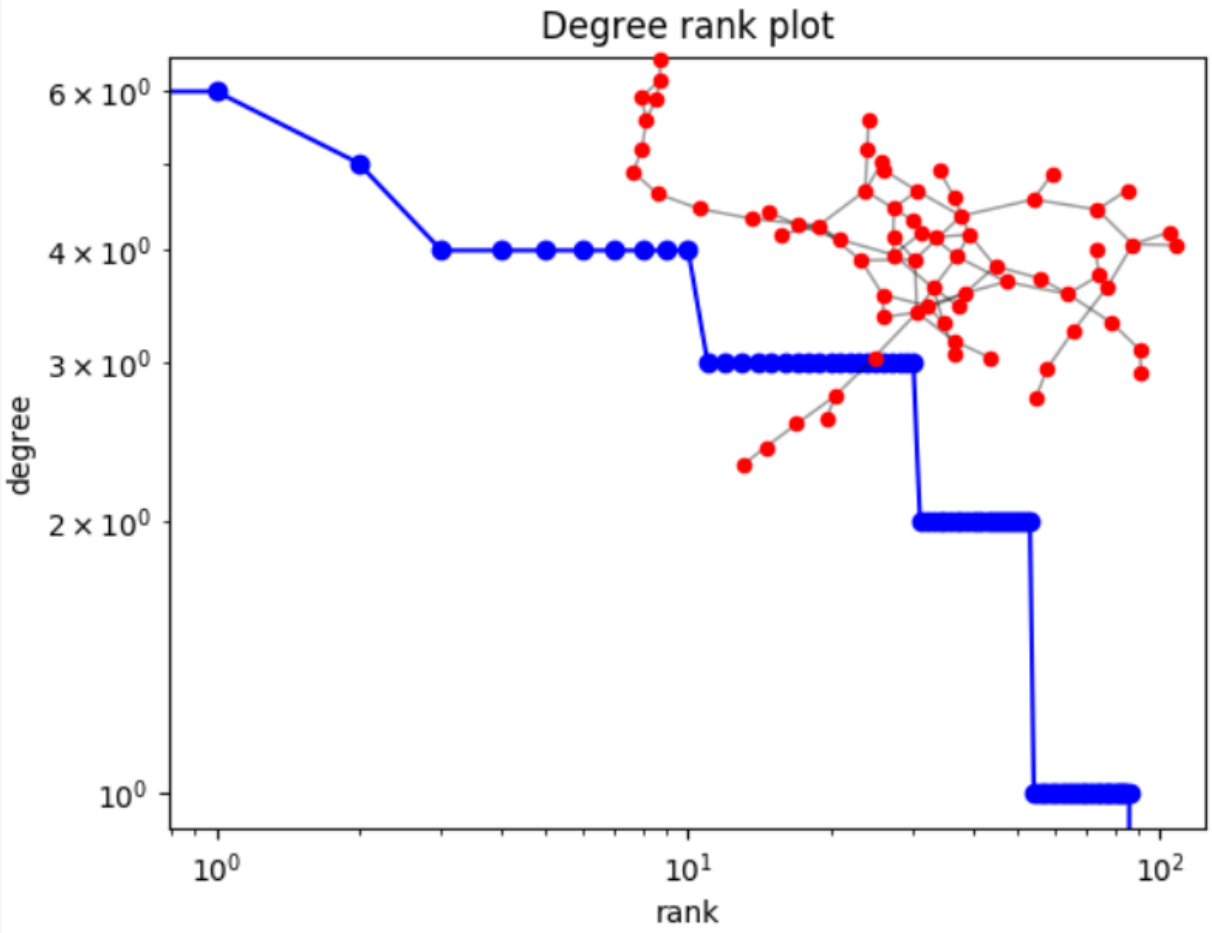
```
import matplotlib.pyplot as plt
import networkx as nx

try:
    import pygraphviz
    from networkx.drawing.nx_agraph import graphviz_layout
except ImportError:
    try:
        import pydot
        from networkx.drawing.nx_pydot import graphviz_layout
    except ImportError:
        raise ImportError("This example needs Graphviz and either "
                          "PyGraphviz or pydot")
```

```
G = nx.balanced_tree(3, 5)
pos = graphviz_layout(G, prog='twopi', args='')
plt.figure(figsize=(8, 8))
nx.draw(G, pos, node_size=20, alpha=0.5, node_color="blue",
with_labels=False)
plt.axis('equal')
plt.show()
```



Degree Rank[edit]

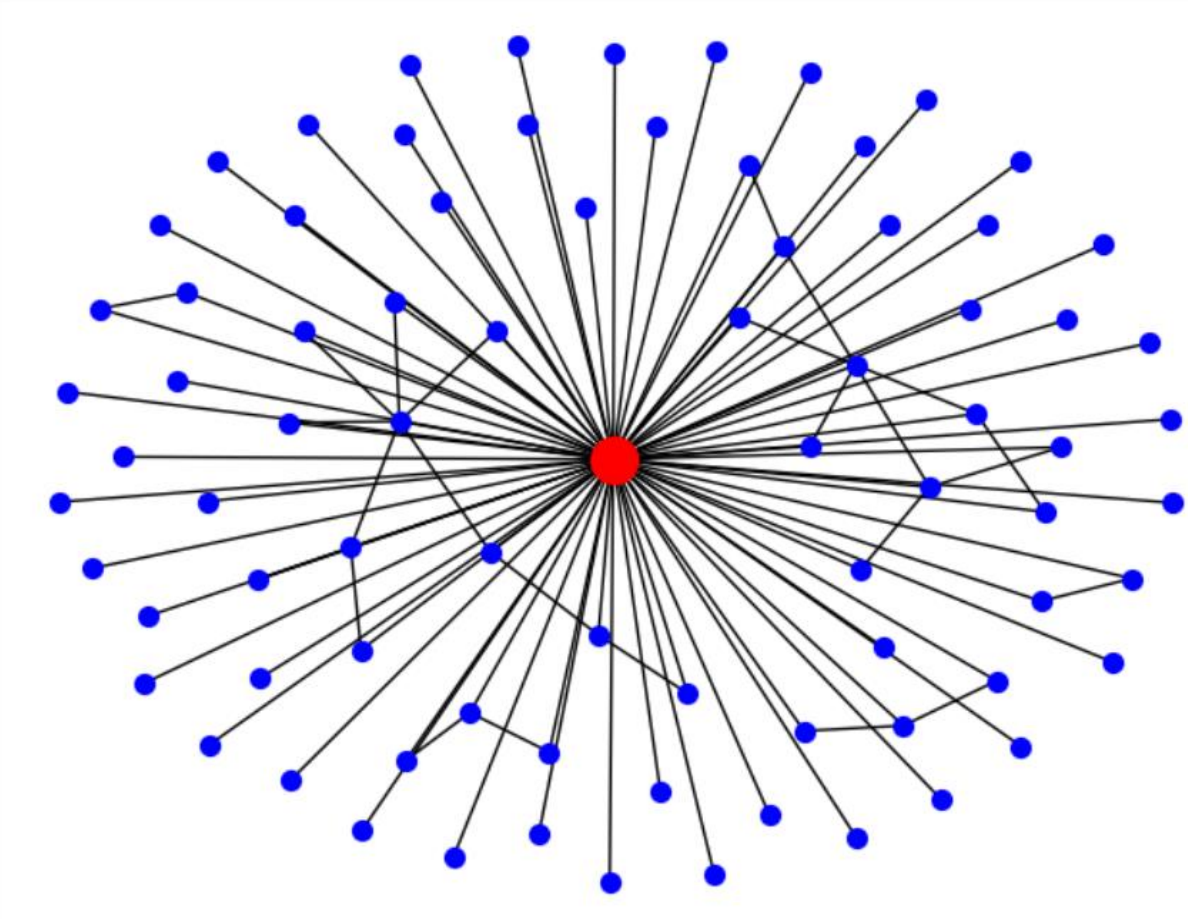
```
import networkx as nx
import matplotlib.pyplot as plt

G = nx.gnp_random_graph(100, 0.02)

degree_sequence = sorted([d for n, d in G.degree()], reverse=True)
# print "Degree sequence", degree_sequence
dmax = max(degree_sequence)

plt.loglog(degree_sequence, 'b-', marker='o')
plt.title("Degree rank plot")
plt.ylabel("degree")
plt.xlabel("rank")

# draw graph in inset
plt.axes([0.45, 0.45, 0.45, 0.45])
Gcc = sorted(nx.connected_component_subgraphs(G), key=len,
reverse=True)[0]
pos = nx.spring_layout(Gcc)
plt.axis('off')
nx.draw_networkx_nodes(Gcc, pos, node_size=20)
nx.draw_networkx_edges(Gcc, pos, alpha=0.4)

plt.show()
```



Ego Graph[edit]

```
from operator import itemgetter

import matplotlib.pyplot as plt
import networkx as nx

if __name__ == '__main__':
    # Create a BA model graph (Barabasi Albert graph)
    n = 1000
    m = 2
    G = nx.generators.barabasi_albert_graph(n, m)
    # find node with largest degree
    node_and_degree = G.degree()
```

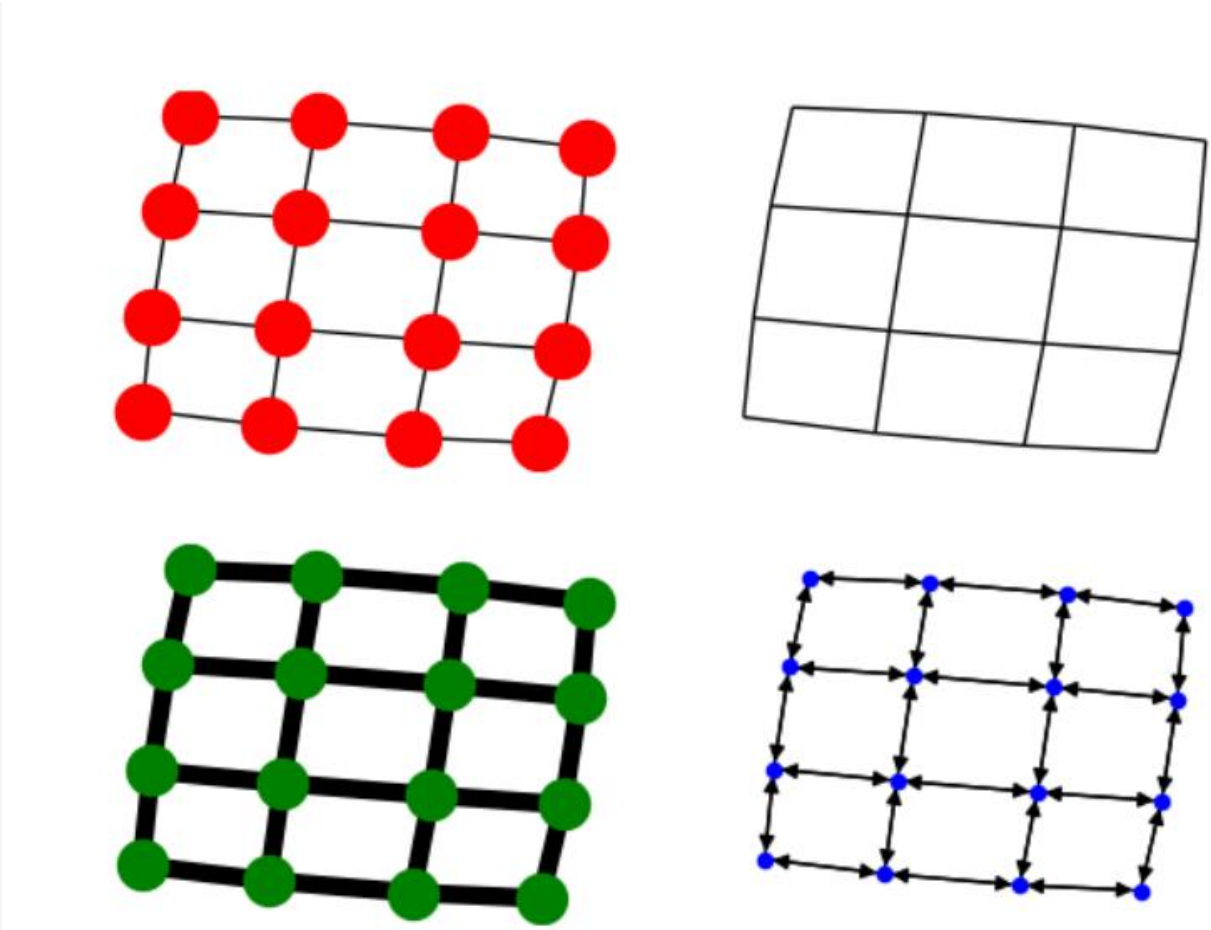
```
    (largest_hub, degree) = sorted(node_and_degree,
key=itemgetter(1))[-1]
    # Create ego graph of main hub
    hub_ego = nx.ego_graph(G, largest_hub)
    # Draw graph
    pos = nx.spring_layout(hub_ego)
    nx.draw(hub_ego, pos, node_color='b', node_size=50,
with_labels=False)
    # Draw ego as large and red
    nx.draw_networkx_nodes(hub_ego, pos, nodelist=[largest_hub],
node_size=300, node_color='r')
    plt.show()
```



Four Grids[edit]

```
import matplotlib.pyplot as plt
import networkx as nx
```

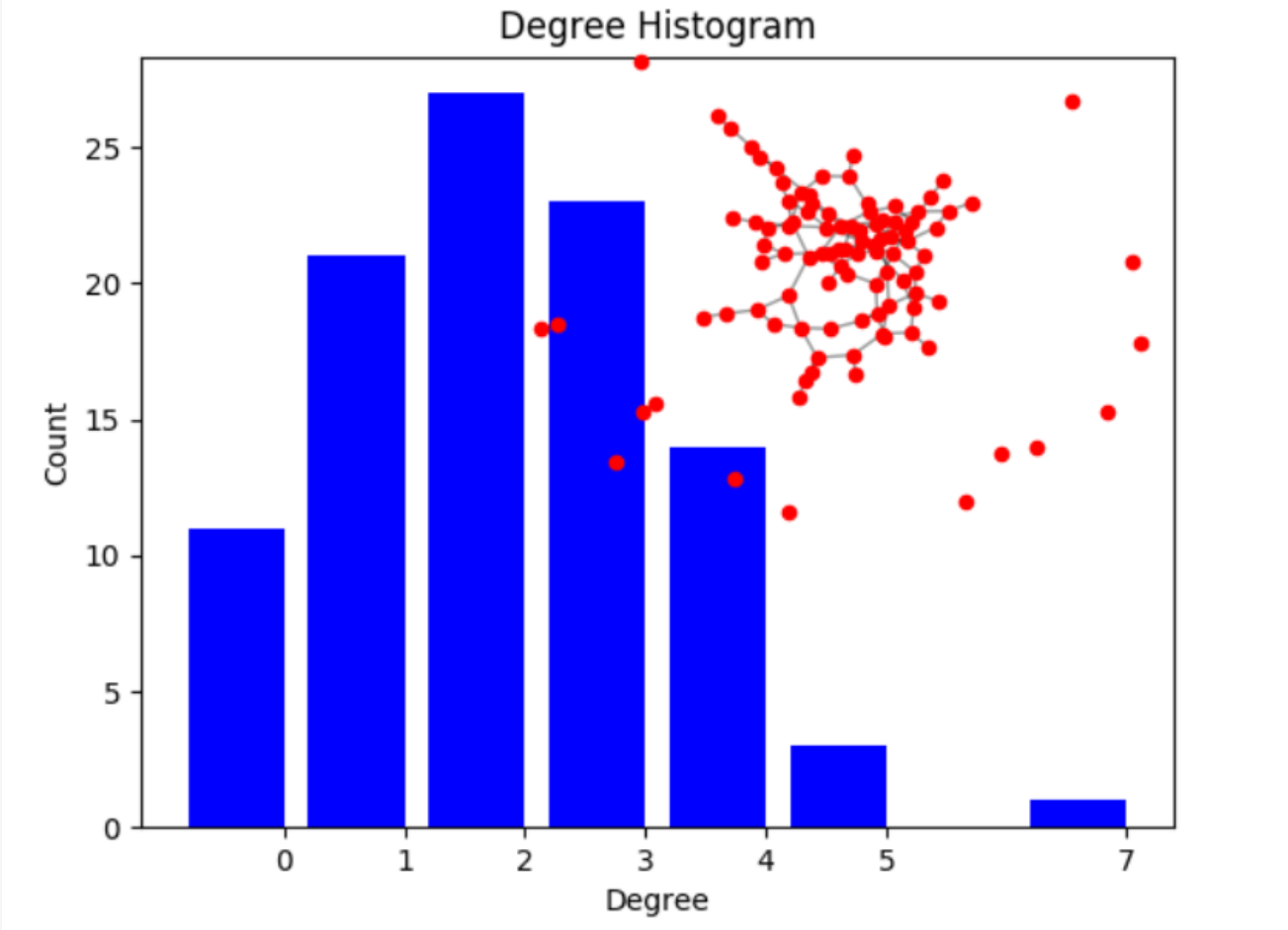
```
G = nx.grid_2d_graph(4, 4)  # 4x4 grid

pos = nx.spring_layout(G, iterations=100)

plt.subplot(221)
nx.draw(G, pos, font_size=8)

plt.subplot(222)
nx.draw(G, pos, node_color='k', node_size=0, with_labels=False)

plt.subplot(223)
nx.draw(G, pos, node_color='g', node_size=250, with_labels=False,
width=6)

plt.subplot(224)
H = G.to_directed()
nx.draw(H, pos, node_color='b', node_size=20, with_labels=False)

plt.show()
```



Degree Histogram[edit]

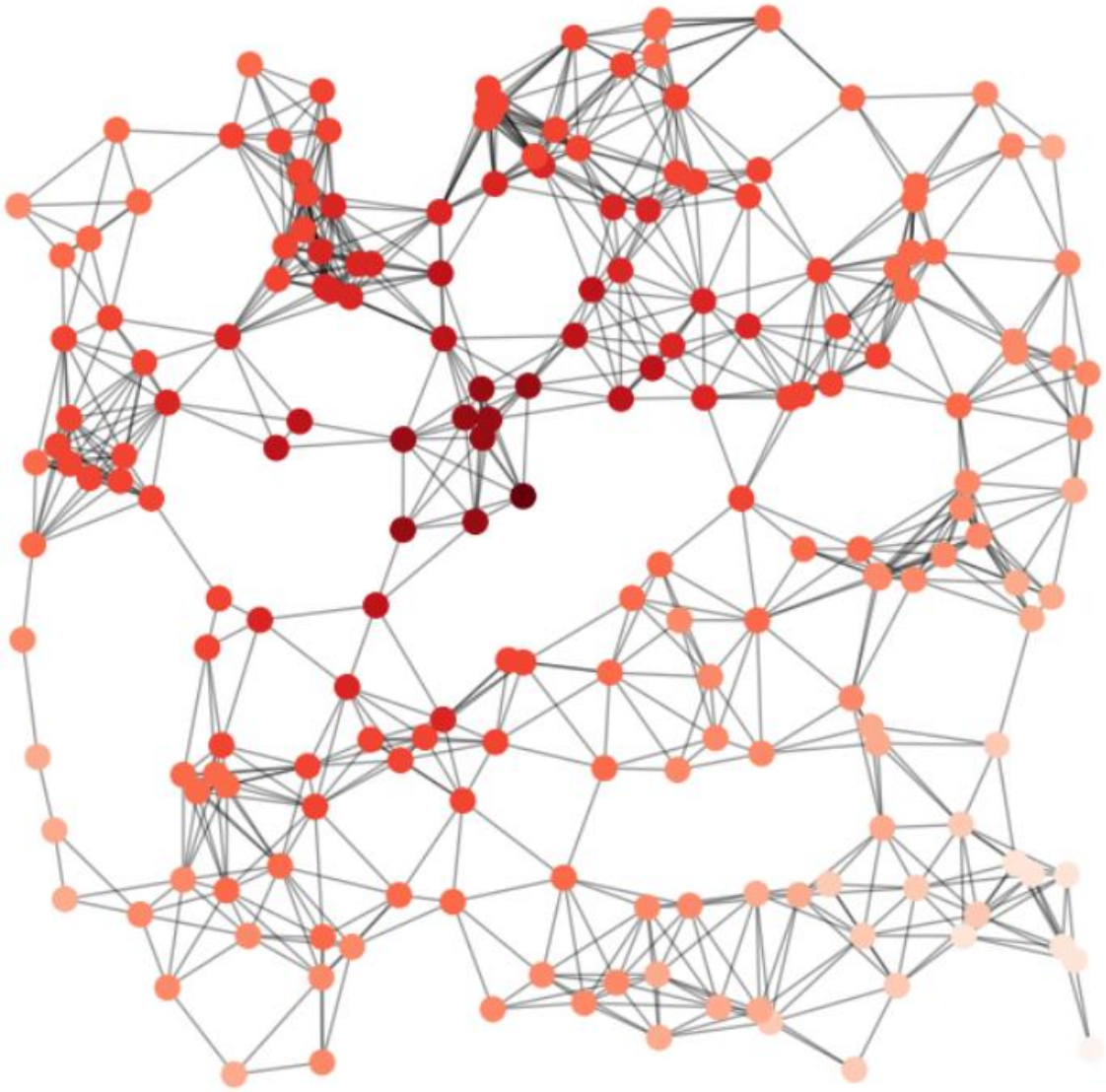
```
import collections
import matplotlib.pyplot as plt
import networkx as nx

G = nx.gnp_random_graph(100, 0.02)

degree_sequence = sorted([d for n, d in G.degree()], reverse=True)  #
degree sequence
# print "Degree sequence", degree_sequence
degreeCount = collections.Counter(degree_sequence)
deg, cnt = zip(*degreeCount.items())

fig, ax = plt.subplots()
plt.bar(deg, cnt, width=0.80, color='b')
```

```
plt.title("Degree Histogram")
plt.ylabel("Count")
plt.xlabel("Degree")
ax.set_xticks([d + 0.4 for d in deg])
ax.set_xticklabels(deg)

# draw graph in inset
plt.axes([0.4, 0.4, 0.5, 0.5])
Gcc = sorted(nx.connected_component_subgraphs(G), key=len,
reverse=True)[0]
pos = nx.spring_layout(G)
plt.axis('off')
nx.draw_networkx_nodes(G, pos, node_size=20)
nx.draw_networkx_edges(G, pos, alpha=0.4)

plt.show()
```



Random Geometric Graph[edit]

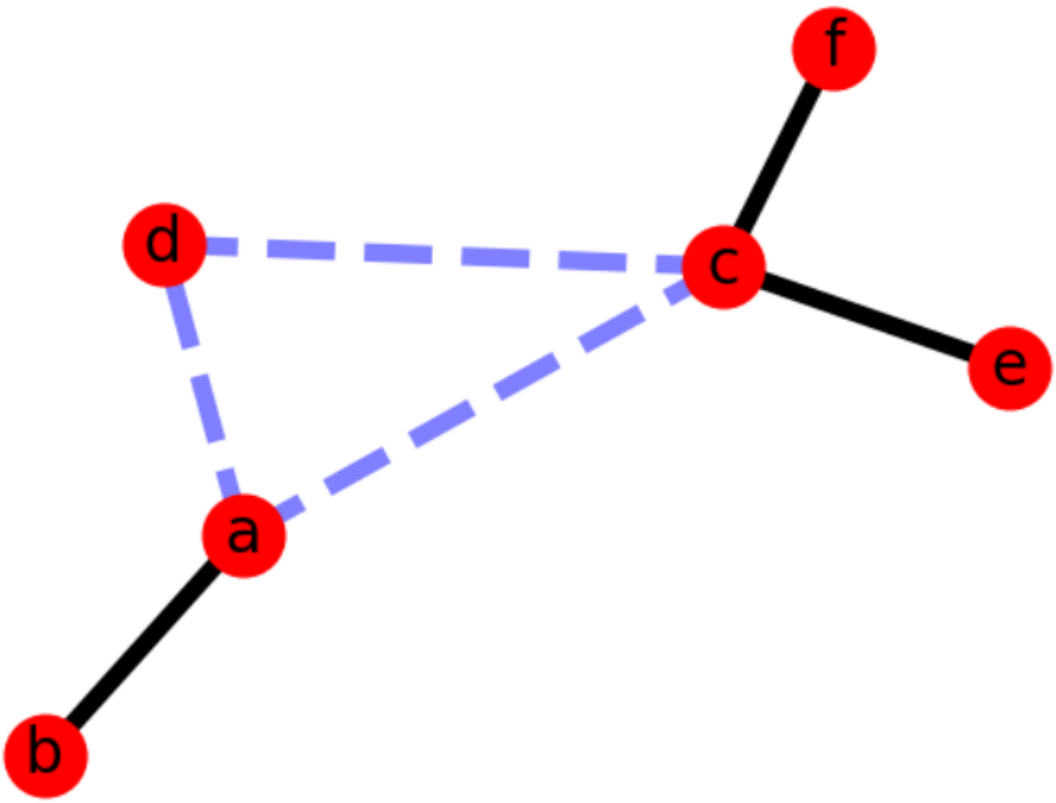
```
import matplotlib.pyplot as plt
import networkx as nx

G = nx.random_geometric_graph(200, 0.125)
# position is stored as node attribute data for random_geometric_graph
pos = nx.get_node_attributes(G, 'pos')

# find node near center (0.5,0.5)
dmin = 1
ncenter = 0
for n in pos:
    x, y = pos[n]
    d = (x - 0.5)**2 + (y - 0.5)**2
    if d < dmin:
        ncenter = n
        dmin = d

# color by path length from node near center
p = dict(nx.single_source_shortest_path_length(G, ncenter))

plt.figure(figsize=(8, 8))
nx.draw_networkx_edges(G, pos, nodelist=[ncenter], alpha=0.4)
nx.draw_networkx_nodes(G, pos, nodelist=list(p.keys()),
                       node_size=80,
                       node_color=list(p.values()),
                       cmap=plt.cm.Reds_r)

plt.xlim(-0.05, 1.05)
plt.ylim(-0.05, 1.05)
plt.axis('off')
plt.show()
```



Weighted Graph[edit]

```
import matplotlib.pyplot as plt
import networkx as nx

G = nx.Graph()

G.add_edge('a', 'b', weight=0.6)
G.add_edge('a', 'c', weight=0.2)
G.add_edge('c', 'd', weight=0.1)
G.add_edge('c', 'e', weight=0.7)
```

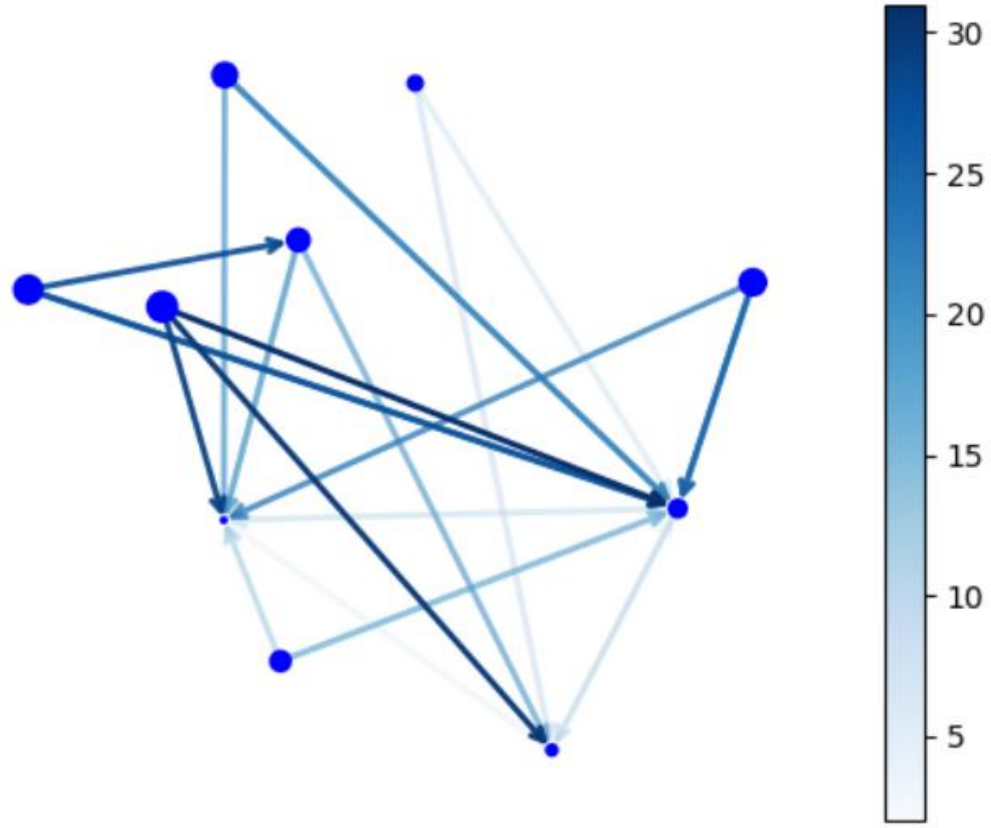
```
G.add_edge('c', 'f', weight=0.9)
G.add_edge('a', 'd', weight=0.3)

elarge = [(u, v) for (u, v, d) in G.edges(data=True) if d['weight'] >
0.5]
esmall = [(u, v) for (u, v, d) in G.edges(data=True) if d['weight'] <=
0.5]

pos = nx.spring_layout(G)  # positions for all nodes

# nodes
nx.draw_networkx_nodes(G, pos, node_size=700)

# edges
nx.draw_networkx_edges(G, pos, edgelist=elarge,
                       width=6)
nx.draw_networkx_edges(G, pos, edgelist=esmall,
                       width=6, alpha=0.5, edge_color='b',
style='dashed')

# labels
nx.draw_networkx_labels(G, pos, font_size=20, font_family='sans-serif')

plt.axis('off')
plt.show()
```



Directed Graph[edit]

```
from __future__ import division
import matplotlib as mpl
import matplotlib.pyplot as plt
import networkx as nx

G = nx.generators.directed.random_k_out_graph(10, 3, 0.5)
pos = nx.layout.spring_layout(G)

node_sizes = [3 + 10 * i for i in range(len(G))]
M = G.number_of_edges()
edge_colors = range(2, M + 2)
edge_alphas = [(5 + i) / (M + 4) for i in range(M)]
```

```
nodes = nx.draw_networkx_nodes(G, pos, node_size=node_sizes,
node_color='blue')
edges = nx.draw_networkx_edges(G, pos, node_size=node_sizes,
arrowstyle='->',
                               arrowsize=10, edge_color=edge_colors,
                               edge_cmap=plt.cm.Blues, width=2)
# set alpha value for each edge
for i in range(M):
    edges[i].set_alpha(edge_alphas[i])

pc = mpl.collections.PatchCollection(edges, cmap=plt.cm.Blues)
pc.set_array(edge_colors)
plt.colorbar(pc)

ax = plt.gca()
ax.set_axis_off()
plt.show()
```

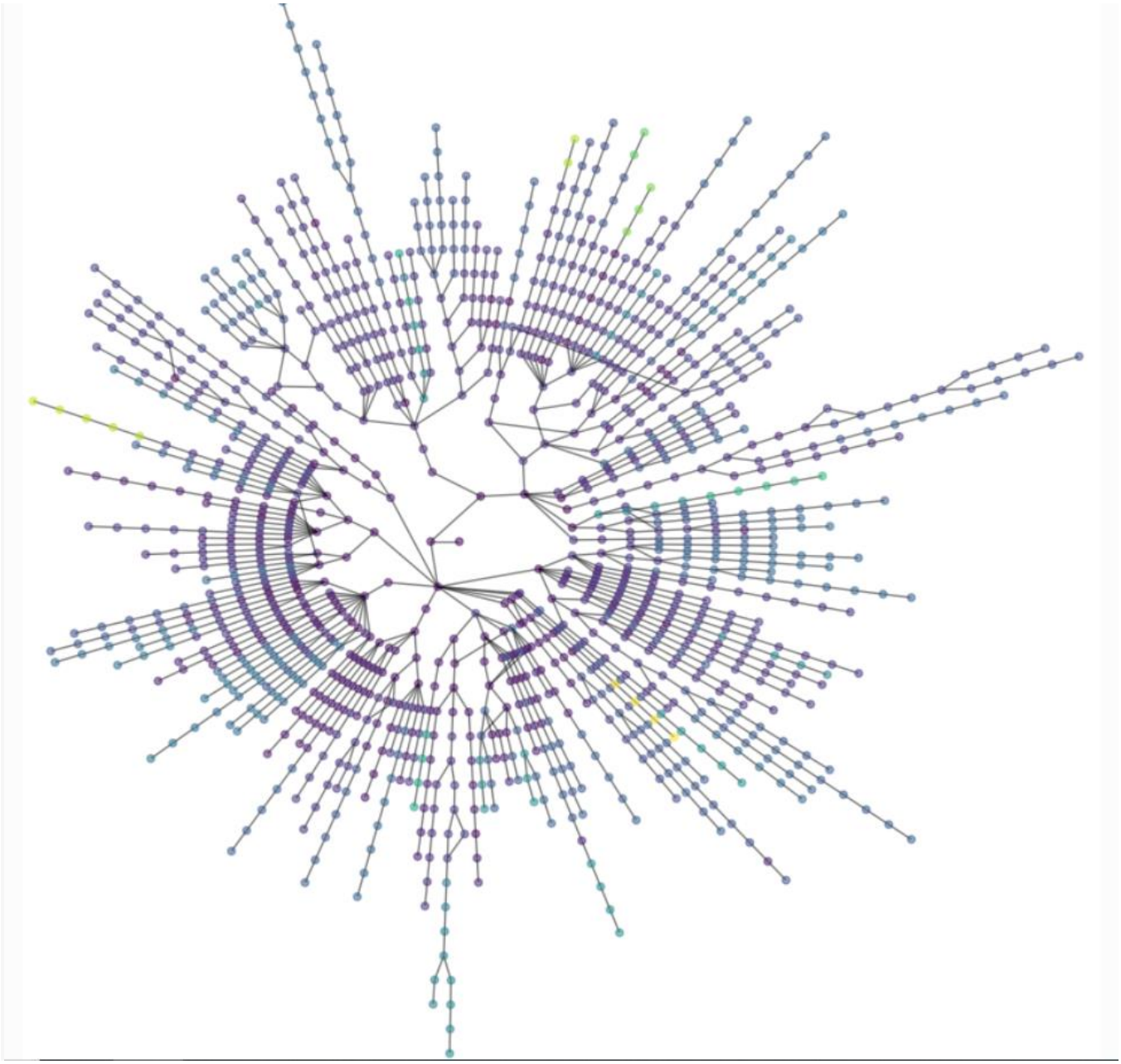


Lanl Routes[edit]

```
import matplotlib.pyplot as plt
import networkx as nx
try:
    import pygraphviz
    from networkx.drawing.nx_agraph import graphviz_layout
except ImportError:
    try:
        import pydot
        from networkx.drawing.nx_pydot import graphviz_layout
    except ImportError:
        raise ImportError("This example needs Graphviz and either "
                          "PyGraphviz or pydot")

def lanl_graph():
    """ Return the lanl internet view graph from lanl.edges
```

```
    """
    try:
        fh = open('lanl_routes.edgelist', 'r')
    except IOError:
        print("lanl.edges not found")
        raise

    G = nx.Graph()

    time = {}
    time[0] = 0  # assign 0 to center node
    for line in fh.readlines():
        (head, tail, rtt) = line.split()
        G.add_edge(int(head), int(tail))
        time[int(head)] = float(rtt)

    # get largest component and assign ping times to G0time dictionary
    G0 = sorted(nx.connected_component_subgraphs(G), key=len,
reverse=True)[0]
    G0.rtt = {}
    for n in G0:
        G0.rtt[n] = time[n]

    return G0


if __name__ == '__main__':

    G = lanl_graph()

    print("graph has %d nodes with %d edges"
          % (nx.number_of_nodes(G), nx.number_of_edges(G)))
    print(nx.number_connected_components(G), "connected components")

    plt.figure(figsize=(8, 8))
    # use graphviz to find radial layout
```

```
    pos = graphviz_layout(G, prog="twopi", root=0)
    # draw nodes, coloring by rtt ping time
    nx.draw(G, pos,
            node_color=[G.rtt[v] for v in G],
            with_labels=False,
            alpha=0.5,
            node_size=15)
    # adjust the plot limits
    xmax = 1.02 * max(xx for xx, yy in pos.values())
    ymax = 1.02 * max(yy for xx, yy in pos.values())
    plt.xlim(0, xmax)
    plt.ylim(0, ymax)
    plt.show()
```



The Erdős-Rényi Random Graph[edit]

During the 1950's the famous mathematician Paul Erdős and Alfred Rényi put forth the concept of a random graph and in the subsequent years of study transformed the world of combinatorics. The random graph is the perfect example of a good mathematical definition: it's simple, has surprisingly intricate structure, and yields many applications.

Definition: Given a positive integer *n* and a probability value *0 <= p <= 1* , define the graph to be the undirected graph *G(n, p)* on *n* vertices whose edges are chosen as follows. For all pairs of vertices *v, w* there is an edge *(v, w)* with probability *p*.

Why Do We Care ?

Random graphs of all sorts (not just Erdős's model) find applications in two very different worlds. The first is pure combinatorics, and the second is in the analysis of networks of all kinds.

The role that random graphs play in this picture is to give us ways to ensure the existence of graphs with certain properties, even if we don't know how to construct an example of such a graph. Indeed, for every theorem proved using random graphs, there is a theorem (or open problem) concerning how to algorithmically construct those graphs which are known to exist.

Random graphs, when they exhibit observed phenomena, have important philosophical consequences. From a bird's-eye view, there are two camps of scientists. The first are those who care about leveraging empirically observed phenomena to solve problems. Many statisticians fit into this realm: they do wonderful magic with data fit to certain distributions, but they often don't know and don't care whether the data they use truly has their assumed properties. The other camp is those who want to discover generative models for the data with theoretical principles. This is more like theoretical physics, where we invent an arguably computational notion of gravity whose consequences explain our observations.

Named Entity Recognition With NLTK[edit]

One of the most major forms of chunking in natural language processing is called "Named Entity Recognition." The idea is to have the machine immediately be able to pull out "entities" like people, places, things, locations, monetary figures, and more.

This can be a bit of a challenge, but NLTK is this built in for us. There are two major options with NLTK's named entity recognition: either recognize all named entities, or recognize named entities as their respective type, like people, places, locations, etc.

Example:

```
import nltk
from nltk.corpus import state_union
from nltk.tokenize import PunktSentenceTokenizer


train_text = state_union.raw("2005-GWBush.txt")
sample_text = state_union.raw("2006-GWBush.txt")


custom_sent_tokenizer = PunktSentenceTokenizer(train_text)


tokenized = custom_sent_tokenizer.tokenize(sample_text)


def process_content():
    try:
        for i in tokenized[5:]:
            words = nltk.word_tokenize(i)
            tagged = nltk.pos_tag(words)
            namedEnt = nltk.ne_chunk(tagged, binary=True)
            namedEnt.draw()
```

```
    except Exception as e:
        print(str(e))



process_content()
```

NE Type and Examples:

- ORGANIZATION - Reliance, WHO
- PERSON - Eddy Bonte, President Obama
- LOCATION - Murray River, Mount Everest
- DATE - June, 2008-06-29
- TIME - two fifty a m, 1:30 p.m.
- MONEY - 175 million Canadian Dollars, GBP 10.40
- PERCENT - twenty percent, 18.75 %

Lecture Notes:Week 10

From JOCWiki

Contents

[hide]

- 1 **NumPy**
- 2 NumPy Arrays
- 3 NumPy basic program
- 4 NumPy array creation methods
- 5 Operations on NumPy array
- 6 Various NumPy Operations
  - 6.1 Creating a Vector
  - 6.2 Creating a Matrix
  - 6.3 Creating a Sparse Matrix
  - 6.4 Selecting Elements
  - 6.5 Describing a Matrix
  - 6.6 Applying Operations to Elements
  - 6.7 Finding the Maximum & Minimum Values
  - 6.8 Calculating Average, Variance & Standard Deviation
  - 6.9 Reshaping Arrays
  - 6.10 Transposing a Vector or a Matrix
  - 6.11 Finding the Determinant & Rank of a Matrix
  - 6.12 Getting the Diagonal of a Matrix
  - 6.13 Calculating the Trace of Matrix
  - 6.14 Finding EigenValues & EigenVectors
  - 6.15 Calculating Dot Products
  - 6.16 Inverting a Matrix
  - 6.17 Generating Random Values
- 7 Flames Game

- 8 Flames Code in Python
- 9 **More About Flames Games**
- 10 **Flames Example**
- 11 Josephus Problem
- 12 Josephus Problem Code in Python
  - 12.1 First Approach
  - 12.2 Alternate Solution with a Circular Linked List
  - 12.3 Recursive Solution
  - 12.4 Dynamic Programming Solution
- 13 String Functions in Python
  - 13.1 Making Strings Uppercase() & Lowercase()
  - 13.2 Boolean Methods
  - 13.3 Determining String Length
  - 13.4 Join Split & Replace Methods
  - 13.5 Example
- 14 Regular Expressions (.*)
  - 14.1 Search & Replace
  - 14.2 Email Extraction Program Using Regular Expressions
- 15 **Image Compression Program**

NumPy [edit]

NumPy is a package for array processing in Python. NumPy provides fast multidimensional array object. It is used for scientific computing. We can perform mathematical and logical operations on arrays.

Installing NumPy: pip install numpy

NumPy Arrays[edit]

We can create a simple array as follows.

import numpy as np x = np.array([1,2,3])

NumPy's array class is called ndarray. The following are the few attributes of ndarray:

ndarray.ndim : returns number of dimensions in the array
ndarray.shape : returns tuple of integers indicating the size of the array in each dimension.
ndarray.size : the total number of elements of the array. This is equal to the product of the elements of ndarray.shape.
ndarray.dtype : describes the data type of the elements in the array.

NumPy basic program[edit]

```
1 # import numpy
2 import numpy as np
3
4 # create array
5 x = np.array( [[ 1, 2, 3],
6                [ 4, 5, 6]] )
```

```
7
8 # type of array object
9 print("type of array is: ", type(x))
10 #output: <class 'numpy.ndarray'>
11
12 # total number of elements in array
13 print("size of array: ", x.size) #output: 6
14
15 # shape of array
16 print("shape of array: ", x.shape)  #output: (2,3)
17
18 # array dimensions
19 print("no. of dimensions: ", x.ndim)  #output: 2
20
21 # data type of elements in array
22 print("the type of elements ", x.dtype) #output: dtype('int32')
```

NumPy array creation methods[edit]

Following are the methods of array creation:

array : to create simple array
zeros : creates array in which elements are only zeros
ones : creates array in which elements are only ones
arange : creates array with elements in increasing order
empty : uninitialized array with random values

Sample program for array creation

```
1 import numpy as np
2
3 a = np.array([1,2,3])     # creates [1 2 3]
4
5 b = np.zeros( (2,3) )
6 '''creates array of zeros
7 [[0. 0. 0.]
8  [0. 0. 0.]]
```

```
'''

c = np.ones( (2,3) )
'''creates array of ones
[[1. 1. 1.]
[1. 1. 1.]]
'''
d = np.arange(10)
'''creates array of values 0 to 9 in increasing order
[0 1 2 3 4 5 6 7 8 9]'''
```

Operations on NumPy array[edit]

The following is a program for simple operations on numpy array such as addition, subtraction, transpose, etc

```
import numpy as np
x = np.array( [[1,2],[3,4]] )
y = np.array( [[4,5],[2,1]] )

print(np.add(x,y)) #addition of x,y
'''
[[5 7]
 [5 5]]
'''
print(np.subtract(x,y)) #subtraction of x,y
'''
[[-3 -3]
 [ 1  3]]
'''
print(np.sqrt(x)) #square root  of all elements of x
'''
[[1.         1.41421356]
 [1.73205081 2.        ]]
'''
```

```
20 print(x.T) #transpose of x
21 '''
22 [[1 3]
23  [2 4]]
24 '''
25 print(np.sum(x,axis = 0)) #column wise sum i.e., [4 6]
26
27 print(np.sum(x,axis = 1)) #row wise sum i.e.,[3 7]
```

Various NumPy Operations[edit]

Numpy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. Moreover, Numpy forms the foundation of the Machine Learning stack.

Creating a Vector[edit]

Here we use Numpy to create a 1-D Array which we then call a vector.

```
#Load Library
import numpy as np

#Create a vector as a Row
vector_row = np.array([1,2,3])

#Create vector as a Column
vector_column = np.array([[1],[2],[3]])
```

Creating a Matrix[edit]

We Create a 2-D Array in Numpy and call it a Matrix. It contains 2 rows and 3 columns.

```
#Load Library
import numpy as np

#Create a Matrix
matrix = np.array([[1,2,3],[4,5,6]])
print(matrix)
```

Creating a Sparse Matrix[edit]

Given data with very few non zero values you want to efficiently represent it. Sparse Matrices store only non zero elements and assume all other values will be zero, leading to significant computational savings.

```
#Load Library
import numpy as np

#Create a Matrix
matrix = np.array([[0,0],[0,1],[3,0]])
print(matrix)

#Create Compressed Sparse Row(CSR) matrix
matrix_sparse = sparse.csr_matrix(matrix)
print(matrix_sparse)
```

Selecting Elements[edit]

When you need to select one or more element in a vector or matrix

```
#Load Library
import numpy as np

#Create a vector as a Row
vector_row = np.array([ 1,2,3,4,5,6 ])

#Create a Matrix
matrix = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(matrix)

#Select 3rd element of Vector
print(vector_row[2])

#Select 2nd row 2nd column
print(matrix[1,1])

#Select all elements of a vector
print(vector_row[:])

#Select everything up to and including the 3rd element
```

```
print(vector_row[:3])

#Select the everything after the 3rd element
print(vector_row[3:])

#Select the last element
print(vector[-1])

#Select the first 2 rows and all the columns of the matrix
print(matrix[:2,:])

#Select all rows and the 2nd column of the matrix
print(matrix[:,1:2])
```

Describing a Matrix[edit]

When you want to know about the shape size and dimensions of a Matrix.

```
import numpy as np



#Create a Matrix
matrix = np.array([[1,2,3],[4,5,6],[7,8,9]])

#View the Number of Rows and Columns
print(matrix.shape)

#View the number of elements (rows*columns)
print(matrix.size)

#View the number of Dimensions(2 in this case)
print(matrix.ndim)
```

Applying Operations to Elements[edit]

You want to apply some function to multiple elements in an array. Numpy's vectorize class converts a function into a function that can apply to multiple elements in an array or slice of an array.

```
#Load Library
```

```
import numpy as np

#Create a Matrix
matrix = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(matrix)

#Create a function that adds 100 to something
add_100 =lambda i: i+100

#Convert it into a vectorized function
vectorized_add_100= np.vectorize(add_100)

#Apply function to all elements in matrix
print(vectorized_add_100(matrix))
```

Finding the Maximum & Minimum Values[edit]

We use Numpy's max and min functions:

```
#Load Library
import numpy as np

#Create a Matrix
matrix = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(matrix)

#Return the max element
print(np.max(matrix))

#Return the min element
print(np.min(matrix))

#To find the max element in each column
print(np.max(matrix,axis=0))

#To find the max element in each row
print(np.max(matrix,axis=1))
```

Calculating Average, Variance & Standard Deviation[edit]

When you want to calculate some descriptive statistics about an array.

```
#Load Library
import numpy as np

#Create a Matrix
matrix = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(matrix)

#Mean
print(np.mean(matrix))

#Standard Dev.
print(np.std(matrix))

#Variance
print(np.var(matrix))
```

Reshaping Arrays[edit]

When you want to reshape an array(changing the number of rows and columns) without changing the elements.

```
#Load Library
import numpy as np

#Create a Matrix
matrix = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(matrix)

#Reshape
print(matrix.reshape(9,1))

#Here -1 says as many columns as needed and 1 row
print(matrix.reshape(1,-1))
```

```
#If we provide only 1 value Reshape would return a 1-d array of that
length
print(marix.reshape(9))

#We can also use the Flatten method to convert a matrix to 1-d array
print(matrix.flatten())
```

Transposing a Vector or a Matrix[edit]

By transposing you interchange the rows and columns of a Matrix.

```
#Load Library
import numpy as np

#Create a Matrix
matrix = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(matrix)

#Transpose the matrix
print(matrix.T)
```

Finding the Determinant & Rank of a Matrix[edit]

The rank of a Matrix is the number of dimensions of the vector space spanned by its rows or columns.

```
#Load Library
import numpy as np

#Create a Matrix
matrix = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(matrix)

#Calculate the Determinant
print(np.linalg.det(matrix))

#Calculate the Rank
print(np.linalg.matrix_rank(matrix))
```

Getting the Diagonal of a Matrix[edit]

When you need to extract only the diagonal elements of a matrix.

```
#Load Library
import numpy as np

#Create a Matrix
matrix = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(matrix)

#Print the Principal diagonal
print(matrix.diagonal())

#Print the diagonal one above the Principal diagonal
print(matrix.diagonal(offset=1))

#Print the diagonal one below Principal diagonal
print(matrix.diagonal(offset=-1))
```

Calculating the Trace of Matrix[edit]

Trace of a Matrix is the sum of elements on the Principal Diagonal of the Matrix.

```
#Load Library
import numpy as np

#Create a Matrix
matrix = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(matrix)

#Print the Trace
print(matrix.trace())
```

Finding EigenValues & EigenVectors[edit]

Eigenvectors are widely used in Machine Learning libraries. Intutively given a linear transformation represented by a matrix,A, eigenvectors are vectors that when that transformation is applied, change only in scale(not direction).

```
#Load Library
import numpy as np
```

```
#Create a Matrix
matrix = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(matrix)

# Calculate the Eigenvalues and Eigenvectors of that Matrix
eigenvalues ,eigenvectors=np.linalg.eig(matrix)
print(eigenvalues)
print(eigenvectors)
```

Calculating Dot Products[edit]

```
#Load Library
import numpy as np

#Create vector-1
vector_1 = np.array([ 1,2,3 ])

#Create vector-2
vector_2 = np.array([ 4,5,6 ])

#Calculate Dot Product
print(np.dot(vector_1,vector_2))

#Alternatively you can use @ to calculate dot products
print(vector_1 @ vector_2)
```

Inverting a Matrix[edit]

This is used when you want to calculate the inverse of a Square Matrix.

```
#Load Library
import numpy as np

#Create a Matrix
matrix = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(matrix)

#Calculate its inverse
```

```
print(np.linalg.inv(matrix))
```

Generating Random Values[edit]

Numpy offers a wide variety of means to generate Random Numbers.

Moreover, It can sometimes be useful to return the same random numbers to get predictable, repeatable results. We can do so by setting the **Seed** (An Integer) of the pseudorandom generator. Random processes with the same seed would always produce the same result.

```
#Load Library
import numpy as np


#Set seed
np.random.seed(1)


#Generate 3 random integers b/w 0 and 10 (start, end, n)
print(np.random.randint(0,11,3))


#Draw 3 numbers from a normal distribution with mean 1.0 and std 2.0
print(np.random.normal(1.0,2.0,3))
```

Flames Game[edit]

Flames is a popular game which is an acronym for Friend, Lover, Affection, Marriage, Enemy, Siblings or Sister. It tells about the relationship status between two persons.

Procedure:

- Take the names of two persons.
- Cancel out the common alphabets in both of their names.
- Then count the number of letters remaining.
- Take the F L A M E S letters.
- Start removing the letter in flames at which the count ends.
- The letter remaining is the end result of the game.

Flames Code in Python[edit]

```
def flames(name1, name2):
    for i in range(len(name1)):
        for j in range(len(name2)):
            if(name1[i]==name2[j]):
                name2=name2.replace(name2[j],"*")
```

```
                break
    #print(name1)
    #print(name2)
    t=name2.count('*')*2
    print(t)
    results = ['friend','love','affection','marriage','enemy','sister']
    t=len(name1)+len(name2)-t-1
    #print(t)
    i=-1
    while(len(results)>1):
        count=-1
        while(count<t):
            count=count+1
            i=(i+1)%len(results)
        results.remove(results[i-1])
    return results[0]

name1 = input('Enter name of first person: ').casefold()
name2 = input('Enter name of second person: ').casefold()
print('Relationship is', flames(name1, name2))
```

More About Flames Games[edit]

FLAMES is a popular game named after the acronym: Friends, Lovers, Affectionate, Marriage, Enemies, Sibling. This game does not accurately predict whether or not an individual is right for you, but it can be fun to play this with your friends.

There are two steps in this game:
Get the count :

- Take the two names.
- Remove the common characters with their respective common occurrences.
- Get the count of the characters that are left .

Get the result :

- Take FLAMES letters as [“F”, “L”, “A”, “M”, “E”, “S”]
- Start removing letter using the count we got.
- The letter which last the process is the result.

Flames Example[edit]

Input : player1 name : AJAY

```
player 2 name : PRIYA
```

Output : Relationship status : Friends
Explaination:
In above given two names A and Y are common letters which are occuring one time(common count) in both names so we are removing these letters from both names. Now count the total letters that are left here it is 5. Now start removing letters one by one from FLAMES using the count we got and the letter which lasts the process is the result. Counting is done in anti-clockwise circular fashion.

FLAMES
counting is start from F, E is at 5th count so we remove E and start counting again but a this time start from S.
FLAMS
M is at 5th count so we remove M and counting start from S.
FLAS
S is at 5th count so we remove S and counting start from F.
FLA
L is at 5th count so we remove L and counting start from A.
FA
A is at 5th count so we remove A. now we have only one letter is remaining so this is the final answer.
F
So, the relationship is F i.e. Friends .

Josephus Problem[edit]

The Josephus problem consists of n number of people that are arranged in a circle. Each person standing in the circle should kill the person next to them and then hand over the sword or a weapon to next living person. This can be started from any person. This procedure continues until only one person is left alive. The last person is the survivor.

Let us say there are six persons in the circle numbered one to six.
123456
If person number 1 starts, then he executes person number 2 and hands over the sword to person number 3.
1x3456
Then person number 3 executes number 4.
1x3x56
Similarly, number 5 executes number 6.
1x3x5x
After that person number 3 is executed by 1.
1xxx5x
and then 1 is executed.
xxxx5x
Only number 5 is remaining.

Josephus Problem Code in Python[edit]

First Approach[edit]

```
>>> def j(n, k):
        p, i, seq = list(range(n)), 0, []
        while p:
                i = (i+k-1) % len(p)
                seq.append(p.pop(i))
        return 'Prisoner killing order: %s.\nSurvivor: %i' % (',
'.join(str(i) for i in seq[:-1]), seq[-1])
```

Output:

```
>>> print(j(5, 2))
Prisoner killing order: 1, 3, 0, 4.
Survivor: 2


>>> print(j(41, 3))
Prisoner killing order: 2, 5, 8, 11, 14, 17, 20, 23, 26, 29, 32, 35,
38, 0, 4, 9, 13, 18, 22, 27, 31, 36, 40, 6, 12, 19, 25, 33, 39, 7, 16,
28, 37, 10, 24, 1, 21, 3, 34, 15.
Survivor: 30
```

Yet another faster way: not showing the killing order

```
>>>def josephus(n, k):
         r = 0
         for i in xrange(1, n+1):
             r = (r+k)%i
         return 'Survivor: %d' %r


>>> print(josephus(5, 2))
Survivor: 2
>>> print(josephus(41, 3))
Survivor: 30
```

Alternate Solution with a Circular Linked List[edit]

The function returns the killing order. The last in the list stays alive. Notice that the result is a permutation of [0, 1, ... n - 1]. In the program, a[p] is the index of the next living prisoner after 'p'. The program stops when p = a[p], that is, when there remains only one living prisoner.

```
def josephus(n, k):
    a = list(range(1, n + 1))
```

```
    a[n - 1] = 0
    p = 0
    v = []
    while a[p] != p:
        for i in range(k - 2):
            p = a[p]
        v.append(a[p])
        a[p] = a[a[p]]
        p = a[p]
    v.append(p)
    return v

josephus(10, 2)
[1, 3, 5, 7, 9, 2, 6, 0, 8, 4]

josephus(41, 3)[-1]
30
```

Recursive Solution[edit]

```
int JosephusHelper(int n,int k)
{
       if(n==1) return 0;  // if there is only one person his index is
0
       return ((JosephusHelper(n-1,k)+k)%n);
}
int Josephus(int n,int k)
{
return 1+JosephusHelper(n,k);  //adding 1 to change indexing
}
```

Dynamic Programming Solution[edit]

```
int Josephus(int n,int k)
{
int result=0; //for n=1
```

```
for(int i=2;i<=n;i++)
{
        result=(k+result)%i;
}
return (result+1);  //adding 1 to change indexing;
}
```

String Functions in Python[edit]

Python has several built-in functions associated with the string data type. These functions let us easily modify and manipulate strings. We can think of functions as being actions that we perform on elements of our code. Built-in functions are those that are defined in the Python programming language and are readily available for us to use.

Making Strings Uppercase() & Lowercase()[edit]

The functions **str.upper()** and **str.lower()** will return a string with all the letters of an original string converted to upper- or lower-case letters. Because strings are immutable data types, the returned string will be a new string. Any characters in the string that are not letters will not be changed.

The **str.upper()** and **str.lower()** functions make it easier to evaluate and compare strings by making case consistent throughout. That way if a user writes their name all lower case, we can still determine whether their name is in our database by checking it against an all upper-case version, for example.

Boolean Methods[edit]

Python has some string methods that will evaluate to a Boolean value. These methods are useful when we are creating forms for users to fill in, for example. If we are asking for a post code we will only want to accept a numeric string, but when we are asking for a name, we will only want to accept an alphabetic string.

- **str.isalnum()**: True if String consists of only alphanumeric characters (no symbols)
- **str.isalpha()**: True if String consists of only alphabetic characters (no symbols)
- **str.islower()**: True if String's alphabetic characters are all lower case
- **str.isnumeric()**: True if String consists of only numeric characters
- **str.isspace()**: True if String consists of only whitespace characters
- **str.istitle()**: True if String is in title case
- **str.isupper()**: True if String's alphabetic characters are all upper case

Using the **str.isnumeric()** method on the string **5** returns a value of **True**, while using the same method on the string **abcdef** returns a value of **False**.

Checking whether characters are lower case, upper case, or title case, can help us to sort our data appropriately, as well as provide us with the opportunity to standardize data we collect by checking and then modifying strings as needed.

Boolean string methods are useful when we want to check whether something a user enters fits within given parameters.

Determining String Length[edit]

The string method **len()** returns the number of characters in a string. This method is useful for when you need to enforce minimum or maximum password lengths, for example, or to truncate larger strings to be within certain limits for use as abbreviations.

Keep in mind that any character bound by single or double quotation marks — including letters, numbers, whitespace characters, and symbols — will be counted by the len() method.

Join Split & Replace Methods[edit]

- The **str.join()** method will concatenate two strings, but in a way that passes one string through another.
- We can also use the **str.join()** method to return a string that is a reversal from the original string.
- The **str.join()** method is also useful to combine a list of strings into a new single string.
- The **str.split()** method returns a list of strings that are separated by whitespace if no other parameter is given.
- We can also use **str.split()** to remove certain parts of an original string.
- The **str.replace()** method can take an original string and return an updated string with some replacement.

Example[edit]

```
import string


text = "Monty Python's Flying Circus"


print "upper", "=>", string.upper(text)
print "lower", "=>", string.lower(text)
print "split", "=>", string.split(text)
print "join", "=>", string.join(string.split(text), "+")
print "replace", "=>", string.replace(text, "Python", "Java")
print "find", "=>", string.find(text, "Python"), string.find(text,
"Java")
print "count", "=>", string.count(text, "n")
```

Output:

```
upper => MONTY PYTHON'S FLYING CIRCUS
lower => monty python's flying circus
split => ['Monty', "Python's", 'Flying', 'Circus']
join => Monty+Python's+Flying+Circus
replace => Monty Java's Flying Circus
find => 6 -1
count => 3
```

Regular Expressions (.*)[edit]

- Regular expressions are a powerful tool for various kinds of string manipulation.
- They are a **Domain specific Language (DSL)** that is present as a library in most of the modern programming languages.

They are useful for two main tasks:

- Verifying that strings match a pattern (for instance, that a string has a format of an email address)
- Performing substitutions in a string.
- Domain Specific Languages are highly specialized mini programming languages.
- Private DSL are often used for specific industrial purposes.

Regular Expressions can be accessed using the **re** module which is a part of the standard library.

After we've defined a regular expression, the **re.match()** function can be used to determine whether it matches at the beginning of the string.

- If it does, match returns an object representing the match, & if not, it returns None.

Raw strings don't escape anything, which makes use of regular expressions easier.

For example,

```
import re


pattern = r"hello"
if re.match(pattern,"hellohellohello"):
     print("Match !")
else:
     print("No match !")
```

- The function **re.search()** finds a match of a pattern anywhere in the string.
- The function **re.findall()** returns a list of all substrings that match a pattern.
- The function **re.finditer()** does the same thing as **re.findall()** except it returns an iterator, rather than a list.

The regex search returns an object with several methods that give details about it.

These methods include **group** which return the string matched, **"Start"** & **"End"** which return the starting and ending positions of the first match and **"Span"** which returns the start and end positions of the first match as a tuple.

```
import re
pattern r"pam"


match = re.search(pattern, "eggspamsausage")
if match:
   print(match.group())
```

```
   print(match.start())
   print(match.end())
   print(match.span())
```

Output:

```
pam
4
7
(4, 7)
```

Search & Replace[edit]

One of the most important "re" methods that use regular expressions is **sub**.

Syntax: re.sub(pattern, repl, string, max = 0)

This method replaces all occurrences of the **pattern** in **string** with **repl**, substituting all occurrences, unless **max** provided. This method returns the modified string.

Email Extraction Program Using Regular Expressions[edit]

To demonstrate a sample usage of regular expressions, we create a program to extract email addresses from a string.

```
import re

pattern = r"([\w\.-]+)@([\w\.-]+(\.[\w\.]+)"
str = "Please contact hello@hotmail.com for assistance"

match = re.search(pattern, str)
if match:
   print(match.group())
```

Output:

```
hello@hotmail.com
```

In case the string contains multiple email addresses, we could use the **re.findall()** method instead of **re.search()** to extract all email addresses.

- **[\w\.-]+** matches one or more word character, dot or dash.
- The regex above says that the string should contain word (with dots and dashes allowed), followed by @ sign, then another similar word, then a dot and another word.

Our regex here contains three groups:

- First part of the email address.
- Domain name without the suffix.
- the domain suffix.

Image Compression Program[edit]

in the program below change the value of comp from any value from 2 ,4,8,16,32 and see the different images quality. you can use any image i used india_map.jpg which i downloaded from google.

```
import numpy
from PIL import Image
im=Image.open("india_map.jpg")
comp=32
pixelmap=im.load()
m=numpy.asanyarray(Image.open('india_map.jpg'))
#print(m) # orignial matirx
img=Image.new(im.mode,im.size)
pixelnew=img.load()

for i in range(img.size[0]):
    for j in range(img.size[1]):
        temp=[1,2,3]
        for k in range(3):
          temp[k]=int(pixelmap[i,j][k]/comp)
        pixelnew[i,j]=tuple(temp)
        #print(pixelnew[i,j])



img.save('india_map1.jpg')
```

j=numpy.asanyarray(Image.open('india_map1.jpg')) #reduced matrix

Lecture Notes:Week 11

From JOCWiki

Contents

[hide]

- 1 Code for Automated FACEBOOK Login

- 2 datetime library
- 3 Classes of datetime
  - 3.1 datetime.date
  - 3.2 datetime.time
  - 3.3 datetime.datetime
  - 3.4 datetime.timedelta
- 4 PYTZ
  - 4.1 Country names with codes
- 5 Calendar
- 6 Draw any Google Icon using Python Turtle
- 7 Speech Recognition Python Program for audio wav files larger than 1 min
- 8 What is Selenium ?
  - 8.1 Example 1
  - 8.2 Example 2
  - 8.3 Example 3
- 9 Install Selenium in Windows
- 10 Why to Choose Python Over Java in Selenium ?
- 11 What is WebDriver ?
- 12 Locating elements Using Selenium WebDriver
  - 12.1 Locating by Id
  - 12.2 Locating by Name
  - 12.3 Locating by XPath
  - 12.4 Locating Hyperlinks by Link Text
  - 12.5 Locating Elements by Tag Name
  - 12.6 Locating Elements by Class Name
  - 12.7 Locating Elements by CSS Selectors
- 13 Special Character
- 14 Browser Automation Whatsapp using Python
- 15 Code for Browser Automation Whatsapp using Python
- 16 Browser Automation Using Selenium
- 17 Installation of Selenium
- 18 Conditions for a year to be leap year
- 19 Code to check year is leap year or not
- 20 Simple Metacharacters
- 21 DateTime Library
  - 21.1 Import datetime library
  - 21.2 Date & Time
  - 21.3 Current Date
  - 21.4 Datetime Time Zone
  - 21.5 Subtract Dates
  - 21.6 Compare Dates
  - 21.7 Formatters
  - 21.8 Date Time Representation
  - 21.9 Date Time Arithmetic
  - 21.10 Date Time Comparison
- 22 Calendar Module
- 23 datetime objects

Code for Automated FACEBOOK Login[edit]

```
Posted By: Som

"""
Created on Wed Apr 17 01:46:40 2019
@author: ssmkhrj
```

```
"""
from selenium import webdriver
import getpass
browser=webdriver.Chrome("/Users/ssmkhrj/Downloads/chromedriver")
browser.get("https://www.facebook.com/")

e = input("Enter your email i.e. linked with facebook:\n")
#Taking the email id from the user and storing it in variable 'e'.
p = getpass.getpass(prompt='Enter your fb password:\n')
#Taking the password from the user and storing it in variable 'e'.
#Note the usage of getpass to keep the password hidden
#But getpass functionality doesn't work on Sypder Console, try Jupyter
Notebook for this.

email = browser.find_element_by_id("email")
email.send_keys(e)
#Sending the email address provided by the user in desired place

password = browser.find_element_by_id("pass")
password.send_keys(p)
#Sending the password provided by the user in desired place

log_in = browser.find_element_by_id("loginbutton")
log_in.click()
#After entering the email-id and password, clicking the login button to
enter
```

datetime library[edit]

The datetime module supplies classes for manipulating dates and times in both simple and complex ways. datetime is a combination of time and date (Month, day, year, hour, second, microsecond). It provides a number of functions to deal with dates, times and time intervals.
date – Manipulate just date (Month, day, year)
time – Time independent of the day (Hour, minute, second, microsecond)

Get current date and time:
a) current time

```
import datetime
datetime_object = datetime.datetime.now()
print(datetime_object)

OR
from datetime import datetime as dt
date_object = dt.today()
print(date_object)

output: 2019-04-08 18:54:42.831086
```

b) Get current date:

```
import datetime

date_object = datetime.date.today()
print(date_object)

output: 2019-04-08
```

c) Date object

```
import datetime

d = datetime.date(2019, 4, 8)
print(d)

output: 2019-04-08
```

Classes of datetime[edit]

datetime.date[edit]

It has three attributes year, month, and day.

Example:

```
from datetime import date

dt = date.today()
print(dt)

print("Current year:", dt.year)
print("Current month:", dt.month)
print("Current day:", dt.day)

# output:
# 2019-04-08
# Current year: 2019
# Current month: 4
# Current day: 8
```

datetime.time[edit]

Its attributes: hour, minute, second, microsecond, and tzinfo.

a) time object
from datetime import time

t = time()

print(t) #output: 00:00:00

t = time(1,56,34,324534) #hour, minute, second, microsecond

print(t) #output: 01:56:34.324534

b) time attributes
from datetime import time

a = time(1, 56, 31)

print("hour: ", a.hour) #hour: 1
print("minute:", a.minute) #minute: 56
print("second:", a.second) #second: 31
print("microsecond:", a.microsecond) #microsecond: 0

datetime.datetime[edit]

A combination of a date and a time. It has following attributes:

- year, month, day, hour, minute, second, microsecond, and tzinfo.

Example:

```
1 from datetime import datetime
2
3 d1 = datetime(2018, 10, 28)  # year, month, day
4 print(d1)                    #2018-10-28 00:00:00
5
6 # year, month, day, hour, minute, second, microsecond
7 d2 = datetime(2018, 10, 28, 9, 40, 59, 322380) # 2018-10-28
09:40:59.322380
8
9 print(d2)
```

datetime.timedelta[edit]

It represents duration or the difference between two dates.

```python
from datetime import datetime, timedelta

current_date = datetime.now()

# future dates
future_date = current_date + timedelta(days=90)

print('Current Date:', current_date)
print('Date after 90 days:', future_date)

# past dates
past_date = current_date - timedelta(days=10)
print('Date before 10 days:', past_date)

# output:
# Current Date: 2019-04-08 19:33:32.856681
# Date after 90 days: 2019-07-07 19:34:28.002737
# Date before 10 days: 2019-03-29 19:33:32.856681
```

PYTZ[edit]

It is an effecient time zone library in Python.

Command for installation: pip install pytz

pytz attributes:
all_timezones: returns a list of all timezones
common_timezones: returns a list of commonly used timezones.

Country names with codes[edit]

```python
import pytz
#country names with their codes
for key, val in pytz.country_names.items():
    print(key, '=', val, end='')
    print()

# output:
```

```
 8 # AD = Andorra
 9 # AE = United Arab Emirates
10 # AF = Afghanistan
11 # ...
12 # ZW = Zimbabwe
```

</br>

Calendar[edit]

The calendar library in Python allows us to display calendars of any year/month and perform calculations on day, month or year.

Example :
1) Calendar of a month :
import calendar

year = 2019

month = 4

print(calendar.month(year, month))

2) Calendar of a year :
import calendar

year = 2019

print(calendar.calendar(year))

3) Check Leap Year with calendar module :

import calendar

print(calendar.isleap(2019)) #returns False

Draw any Google Icon using Python Turtle[edit]

```
#download any  icon from google images and save it upside down into a
file called my2.jpg
#for better clarity decrease the inc value, for speed of drawing
increase the inc value
#Good for drawing icons of size 200X200 pixels. it will take more time
if you increase the picture size


inc=5  # change the value for pixel size
image_name="my2.jpg"  # name of the image to be converted
```

```
from PIL import Image
import turtle
import time
import tkinter as tk



a=turtle.Turtle()
turtle.bgcolor("pink")

# Open our image
im = Image.open(image_name)

rgb_im = im.convert('RGB')

width = rgb_im.size[0]
height = rgb_im.size[1]
turtle.setup(width*2,height*2)
print( height,width)
#a.screen.screensize(height,width)
#a.screen.setworldcoordinates(0,0,height,width)
# set some counters for current row and column and total pixels
row = 1
col = 1
pix = 0

hint=-height/2
wint=-width/2
a.penup()
a.goto(wint,hint)
a.pendown()
# create an empty output row
rowdata = ""
while row < height + 1:
    #print("")
    #print("Row number: " + str(row))
```

```
    while col < width + 1:

        r, g, b = rgb_im.getpixel((col-1,row-1))
        a.screen.colormode(255)
        a.pencolor(r,g,b)
        a.pensize(inc)
        a.speed('fastest')
        a.pendown()
        #print(col-inc,row-inc)
        a.goto(col+wint,row+hint)
        a.penup()

        col = col + inc
        # increment the pixel count
        pix = pix + inc

    rowdata = ""
    # increment the row...
    row = row + inc
    # reset the column count
    col = 1
    a.penup()
    a.goto(wint,row+hint-inc)
    print(row-inc)
print("completed execution")

b=input("enter any character to clear turtle screen and continue")
#a.reset()
print("Turtle ready for next program")
```

Speech Recognition Python Program for audio wav files larger than 1 min[edit]

```
# if you use it for more than one hour, google will automatically
# removes your access and hence if you want to do it you may
```

```
# need to purchase the feature

import speech_recognition as sr
import time

sleep_time=2 # in seconds >0
total_duration=210 # total duration of the file in seconds
offset=0 # start from the beginning of the file
duration=30 # cannot be greater than 60 seconds

if total_duration%duration != 0:

 N = int(total_duration/duration) + 2  # No of trails

 print("\n\n",N)
else:

   N = int(total_duration/duration) + 1
for i in range(1,N):

 Audio_file= "i.wav" # Name of the audio file in Wav format
 r=sr.Recognizer()
 with sr.AudioFile(Audio_file) as source:
   audio=r.record(source,offset=offset,duration = duration)
   time.sleep(sleep_time)

 try:
   print(end="" + r.recognize_google(audio))

 except sr.UnknownValueError:
    print("\n\n\nAudio file failed after ",(i-1)*duration, "seconds to
convert to text, Try again \n \n\n" )
    i=i-1
    continue
 finally:
      offset=offset+duration
```

```
print("\n\nCompleted the converstion \n\n")
```

What is Selenium ?[edit]

Selenium is a tool to test your web application.

You can do this in various ways, for instance

- Permit it to tap on buttons
- Enter content in structures
- Skim your site to check whether everything is "OK" and so on.

Key Points

- Selenium is an open-source web-based automation tool.
- Python language is used with Selenium for testing. It has far less verbose and easy to use than any other programming language
- The Python APIs empower you to connect with the browser through Selenium
- Selenium can send the standard Python commands to different browsers, despite variation in their browser's design.

Example 1[edit]

- Open a new Firefox browser
- Load the page at the given URL

```
from selenium import webdriver


browser = webdriver.Firefox()

browser.get('http://seleniumhq.org/')
```

Example 2[edit]

- Open a new Firefox browser
- Load the Yahoo homepage
- Search for “seleniumhq”
- Close the browser

```
from selenium import webdriver

from selenium.webdriver.common.keys import Keys


browser = webdriver.Firefox()


browser.get('http://www.yahoo.com')

assert 'Yahoo' in browser.title


elem = browser.find_element_by_name('p')  # Find the search box
```

```
elem.send_keys('seleniumhq' + Keys.RETURN)


browser.quit()
```

Example 3[edit]

Selenium WebDriver is often used as a basis for testing web applications. Here is a simple example using Python's standard unittest library:

```
import unittest
from selenium import webdriver


class GoogleTestCase(unittest.TestCase):


    def setUp(self):
        self.browser = webdriver.Firefox()
        self.addCleanup(self.browser.quit)


    def testPageTitle(self):
        self.browser.get('http://www.google.com')
        self.assertIn('Google', self.browser.title)


if __name__ == '__main__':
    unittest.main(verbosity=2)
```

Install Selenium in Windows[edit]

```
1. How to install selenium
    http://sccilabs.org/jocwiki/index.php/File:Install_selenium.mp4
2. Copy Selenium to Anaconda
    http://sccilabs.org/jocwiki/index.php/File:Copy_Selenium.mp4
3. Find Chrome driver location on your system


http://sccilabs.org/jocwiki/index.php/File:Chrome_Driver_file_location.
mp4
```

Why to Choose Python Over Java in Selenium ?[edit]

Few points that favor Python over Java to use with Selenium is,

- Java programs tend to run slower compared to Python programs.
- Java uses traditional braces to start and ends blocks, while Python uses indentation.
- Java employs static typing, while Python is dynamically typed.
- Python is simpler and more compact compared to Java.

What is WebDriver ?[edit]

WebDriver is a web automation framework that allows you to execute your tests against different browsers.

WebDriver also enables you to use a programming language in creating your test scripts. We can also use conditional operations like if-then-else or switch-case. We can also perform looping controls like do-while.

WebDriver's Architecture is Simpler

- It controls the browser from the OS level
- All you need are your programming language's IDE (which contains your Selenium commands) and a browser.

Speed

- WebDriver is faster since it speaks directly to the browser uses the browser's own engine to control it.

Real-Life Interaction

- WebDriver interacts with page elements in a more realistic way. For example, if you have a disabled text box on a page you were testing, WebDriver really cannot enter any value in it just as how a real person cannot.

Browser Support

- WebDriver can support the headless HtmlUnit browser
- HtmlUnit is termed as "headless" because it is an invisible browser - it is GUI-less.
- It is a very fast browser because no time is spent in waiting for page elements to load. This accelerates your test execution cycles. Since it is invisible to the user, it can only be controlled through automated means.

Locating elements Using Selenium WebDriver[edit]

There are various strategies to locate elements in a page. You can use the most appropriate one for your case. Selenium provides the following methods to locate elements in a page:

- find_element_by_id
- find_element_by_name
- find_element_by_xpath
- find_element_by_link_text
- find_element_by_partial_link_text

- find_element_by_tag_name
- find_element_by_class_name
- find_element_by_css_selector

To find multiple elements (these methods will return a list):

- find_elements_by_name
- find_elements_by_xpath
- find_elements_by_link_text
- find_elements_by_partial_link_text
- find_elements_by_tag_name
- find_elements_by_class_name
- find_elements_by_css_selector

Apart from the public methods given above, there are two private methods which might be useful with locators in page objects. These are the two private methods: **find_element** and **find_elements**.

These are the attributes available for By class:

- ID = "id"
- XPATH = "xpath"
- LINK_TEXT = "link text"
- PARTIAL_LINK_TEXT = "partial link text"
- NAME = "name"
- TAG_NAME = "tag name"
- CLASS_NAME = "class name"
- CSS_SELECTOR = "css selector"

For Example

```
from selenium.webdriver.common.by import By


driver.find_element(By.XPATH, '//button[text()="Some text"]')
driver.find_elements(By.XPATH, '//button')
```

Locating by Id[edit]

Use this when you know *id* attribute of an element. With this strategy, the first element with the id attribute value matching the location will be returned. If no element has a matching *id* attribute, a **NoSuchElementException** will be raised.

Locating by Name[edit]

Use this when you know *name* attribute of an element. With this strategy, the first element with the *name* attribute value matching the location will be returned. If no element has a matching *name* attribute, a **NoSuchElementException** will be raised.

Locating by XPath[edit]

XPath is the language used for locating nodes in an XML document. As HTML can be an implementation of XML (XHTML), Selenium users can leverage this powerful language to target elements in their web applications. XPath extends beyond (as well as supporting) the simple methods of locating by id or name attributes, and opens up all sorts of new possibilities such as locating the third checkbox on the page.

One of the main reasons for using XPath is when you don't have a suitable id or name attribute for the element you wish to locate. You can use XPath to either locate the element in absolute terms (not advised), or relative to an element that does have an id or name attribute. XPath locators can also be used to specify elements via attributes other than id and name.

Absolute XPaths contain the location of all elements from the root (html) and as a result are likely to fail with only the slightest adjustment to the application. By finding a nearby element with an id or name attribute (ideally a parent element) you can locate your target element based on the relationship. This is much less likely to change and can make your tests more robust.

Locating Hyperlinks by Link Text[edit]

Use this when you know *link* text used within an anchor tag. With this strategy, the first element with the link text value matching the location will be returned. If no element has a matching link text attribute, a **NoSuchElementException** will be raised.

Locating Elements by Tag Name[edit]

Use this when you want to locate an element by *tag name*. With this strategy, the first element with the given tag name will be returned. If no element has a matching tag name, a **NoSuchElementException** will be raised.

Locating Elements by Class Name[edit]

Use this when you want to locate an element by *class attribute name*. With this strategy, the first element with the matching class attribute name will be returned. If no element has a matching *class attribute name*, a **NoSuchElementException** will be raised.

Locating Elements by CSS Selectors[edit]

Use this when you want to locate an element by CSS selector syntax. With this strategy, the first element with the matching CSS selector will be returned. If no element has a matching CSS selector, a **NoSuchElementException** will be raised.

Special Character[edit]

```
1 import re
2 string=input()
3 def run(string):
4     regex = re.compile('[@_!#$%^&*()<>?/\|}{~:]')
5     if(regex.search(string) == None):
6         print("NO")
7 
8     else:
```

```
        print("YES")
run(string)
```

Browser Automation Whatsapp using Python[edit]

Have you ever wished to automatically wish your friends on their birthdays, or send a set of messages to your friend (or any Whastapp contact!) automatically at a pre-set time, or send your friends by sending thousands of random text on whatsapp! Using Browser Automation you can do all of it and much more! First you must install these:-

1) Python Bindings for Selenium (Browser Automation software)

pip install selenium

2) Chrome webdriver

Download Chrome driver

3) Chromium Web Browser(Open source version of chrome browser)

sudo apt-get install chromium-browser

Code for Browser Automation Whatsapp using Python[edit]

```
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
import time

# Replace below path with the absolute path
# to chromedriver in your computer
driver = webdriver.Chrome('/home/Divya/Downloads/chromedriver')

driver.get("https://web.whatsapp.com/")
wait = WebDriverWait(driver, 600)

# Replace 'Friend's Name' with the name of your friend
# or the name of a group
target = '"Friend\'s Name"'
```

```
18
19 # Replace the below string with your own message
20 string = "Message sent using Python!!!"
21
22 x_arg = '//span[contains(@title,' + target + ')]'
23 group_title = wait.until(EC.presence_of_element_located((
24     By.XPATH, x_arg)))
25 group_title.click()
26 inp_xpath = '//div[@class="input"][@dir="auto"][@data-tab="1"]'
27 input_box = wait.until(EC.presence_of_element_located((
28     By.XPATH, inp_xpath)))
29 for i in range(100):
30     input_box.send_keys(string + Keys.ENTER)
31     time.sleep(1)
```

Browser Automation Using Selenium[edit]

Selenium is a powerful tool for controlling web browser through program. It is functional for all browsers, works on all major OS and its scripts are written in various languages i.e Python, Java, C# etc, we will be working with Python.

Mastering Selenium will help you automate your day to day tasks like controlling your tweets, Whatsapp texting and even just googling without actually opening a browser in just 15-30 lines of python code. The limits of automation is endless with selenium.

Installation of Selenium[edit]

1.1 Selenium Bindings in Python
Selenium Python bindings provide a convenient API to access Selenium Web Driver like Firefox,Chrome,etc.

Pip install Selenium

1.2 Web Drivers
Selenium requires a web driver to interface with the chosen browser.Web drivers is a package to interact with web browser. It interacts with the web browser or a remote web server through a wire protocol which is common to all. You can check out and install the web drivers of your browser choice.
Chrome: https://sites.google.com/a/chromium.org/chromedriver/downloads
Firefox: https://github.com/mozilla/geckodriver/releases
Safari: https://webkit.org/blog/6900/webdriver-support-in-safari-10/

Conditions for a year to be leap year[edit]

These are the conditions for a leap year :

- The year is evenly divisible by 4;
- If the year can be evenly divided by 100, it is NOT a leap year, unless;
- The year is also evenly divisible by 400. Then it is a leap year.

Code to check year is leap year or not[edit]

```
1 year=int(input("Enter year to be checked:"))
2 if(year%4==0 and year%100!=0 or year%400==0):
3     print("The year is a leap year!)
4 else:
5     print("The year isn't a leap year!)
```

Simple Metacharacters[edit]

Metacharacters are what make regular expressions more powerful than normal string methods. They allow us to create regular expressions to represent concepts like "one or more repetitions" of a vowel.

The existence of metacharacters poses a problem if we want to create a regular expression (or regex) that matches a literal metacharacter such as "$". We can do this by escaping the metacharacters by putting a backslash in front of them.

To avoid this, we can use a raw string, which is a normal string with an "r" in front of it.

- **.(dot)** matches any character other than a anew line.
- ^ and **$** match the **start** and **end** of a string respectively.

DateTime Library[edit]

Import datetime library[edit]

Python datetime functions are provided by the library named datetime and in order to use date time functions we need to import this library like below:

from datetime import datetime

Date & Time[edit]

While using pythons datetime functions, libraries and data structures we will involve with two basic data structures date and time. Date part or object is used to hold date. Time part or object is used to hold time. We can use these separately without providing other part.

Current Date[edit]

Current date or simply now represents date and time we are currently. We will use now function in order to get current date and time. now is provided by datetime object. **now**provides current date and time like below:

```
datetime.today()
```

```
now=datetime.now()
```

We see that date time returns following values:

- Year
- Month
- Date
- Hour
- Minute
- Second

Datetime Time Zone[edit]

Every location in the world have different time but in order to make things more properly time zones are created. Time zones are used to synchronize some area time to the same time. The time zone may change according to location. We can get time zone information with **tzinfo** command like below.:

```
now=datetime.now()
```

```
now.tzinfo
```

Subtract Dates[edit]

Dates can be also subtracted from each other. This is very useful feature to find the interval between two dates as years, months, days, hours, minutes, seconds.

```
t1=datetime.now()
```

```
t2=datetime.now()
```

```
t2-t1
```

Compare Dates[edit]

We can use normal logical comparison operators like > , < and =

```
t1=datetime.now()
```

```
t2=datetime.now()
```

```
if(t1<t2):
```

```
    print("t1 is lower than t2")
```

Formatters[edit]

We can use following formatters

- **%a** Locale’s abbreviated weekday name.

- **%A** Locale’s full weekday name.
- **%b** Locale’s abbreviated month name.
- **%B** Locale’s full month name.
- **%c** Locale’s appropriate date and time representation.
- **%d** Day of the month as a decimal number [01,31].
- **%f** Microsecond as a decimal number [0,999999], zero-padded on the left
- **%H** Hour (24-hour clock) as a decimal number [00,23].
- **%I** Hour (12-hour clock) as a decimal number [01,12].
- **%j** Day of the year as a decimal number [001,366].
- **%m** Month as a decimal number [01,12].
- **%M** Minute as a decimal number [00,59].
- **%p** Locale’s equivalent of either AM or PM.
- **%S** Second as a decimal number [00,61].
- **%U** Week number of the year (Sunday as the first day of the week)
- **%w** Weekday as a decimal number [0(Sunday),6].
- **%W** Week number of the year (Monday as the first day of the week)
- **%x** Locale’s appropriate date representation.
- **%X** Locale’s appropriate time representation.
- **%y** Year without century as a decimal number [00,99].
- **%Y** Year with century as a decimal number.
- **%z** UTC offset in the form +HHMM or -HHMM.
- **%Z** Time zone name (empty string if the object is naive).
- **%%** A literal ‘%’ character.

Date Time Representation[edit]

A date and its various parts are represented by using different datetime functions. Also, there are format specifiers which play a role in displaying the alphabetical parts of a date like name of the month or week day.

For Example,

```
import datetime
print 'The Date Today is  :', datetime.datetime.today()

date_today = datetime.date.today()
print date_today
print 'This Year   :', date_today.year
print 'This Month    :', date_today.month
print 'Month Name:',date_today.strftime('%B')
print 'This Week Day    :', date_today.day
print 'Week Day Name:',date_today.strftime('%A')
```

When we execute the above code, it produces the following result.

```
The Date Today is  : 2018-04-22 15:38:35.835000
2018-04-22
This Year   : 2018
```

```
This Month    : 4
Month Name: April
This Week Day    : 22
Week Day Name: Sunday
```

Date Time Arithmetic[edit]

For calculations involving dates we store the various dates into variables and apply the relevant mathematical operator to these variables.

```
import datetime

#Capture the First Date
day1 = datetime.date(2018, 2, 12)
print 'day1:', day1.ctime()

# Capture the Second Date
day2 = datetime.date(2017, 8, 18)
print 'day2:', day2.ctime()

# Find the difference between the dates
print 'Number of Days:', day1-day2



date_today  = datetime.date.today()

# Create a delta of Four Days
no_of_days = datetime.timedelta(days=4)

# Use Delta for Past Date
before_four_days = date_today - no_of_days
print 'Before Four Days:', before_four_days

# Use Delta for future Date
after_four_days = date_today + no_of_days
print 'After Four Days:', after_four_days
```

When we execute the above code, it produces the following result.

```
day1: Mon Feb 12 00:00:00 2018
day2: Fri Aug 18 00:00:00 2017
Number of Days: 178 days, 0:00:00
Before Four Days: 2018-04-18
After Four Days: 2018-04-26
```

Date Time Comparison[edit]

Date and time are compared using logical operators. But we must be careful in comparing the right parts of the dates with each other.

```
import datetime

date_today  = datetime.date.today()

print 'Today is: ', date_today
# Create a delta of Four Days
no_of_days = datetime.timedelta(days=4)

# Use Delta for Past Date
before_four_days = date_today - no_of_days
print 'Before Four Days:', before_four_days

after_four_days =  date_today + no_of_days

date1 = datetime.date(2018,4,4)

print 'date1:',date1

if date1 == before_four_days :
    print 'Same Dates'
if date_today > date1:
    print 'Past Date'
if date1 < after_four_days:
    print 'Future Date'
```

Calendar Module[edit]

class calendar.Calendar([firstweekday])

- Creates a Calendar object.
- firstweekday is an integer specifying the first day of the week. 0 is Monday (the default), 6 is Sunday.
- A Calendar object provides several methods that can be used for preparing the calendar data for formatting. This class doesn't do any formatting itself. This is the job of subclasses.

Calendar instances have the following methods:

- **iterweekdays()**
- Return an iterator for the week day numbers that will be used for one week. The first value from the iterator will be the same as the value of the firstweekday property.
- **itermonthdates(year, month)**
- Return an iterator for the month month (1–12) in the year year. This iterator will return all days (as datetime.date objects) for the month and all days before the start of the month or after the end of the month that are required to get a complete week.
- **itermonthdays2(year, month)**
- Return an iterator for the month month in the year year similar to itermonthdates(). Days returned will be tuples consisting of a day number and a week day number.
- **itermonthdays(year, month)**
- Return an iterator for the month month in the year year similar to itermonthdates(). Days returned will simply be day numbers.
- **monthdatescalendar(year, month)**
- Return a list of the weeks in the month month of the year as full weeks. Weeks are lists of seven datetime.date objects.
- **monthdays2calendar(year, month)**
- Return a list of the weeks in the month month of the year as full weeks. Weeks are lists of seven tuples of day numbers and weekday numbers.
- **monthdayscalendar(year, month)**
- Return a list of the weeks in the month month of the year as full weeks. Weeks are lists of seven day numbers.
- **yeardatescalendar(year[, width])**
- Return the data for the specified year ready for formatting. The return value is a list of month rows. Each month row contains up to width months (defaulting to 3). Each month contains between 4 and 6 weeks and each week contains 1–7 days. Days are datetime.date objects.
- **yeardays2calendar(year[, width])**
- Return the data for the specified year ready for formatting (similar to yeardatescalendar()). Entries in the week lists are tuples of day numbers and weekday numbers. Day numbers outside this month are zero.
- **yeardayscalendar(year[, width])**
- Return the data for the specified year ready for formatting (similar to yeardatescalendar()). Entries in the week lists are day numbers. Day numbers outside this month are zero.

datetime objects[edit]

A datetime object is a single object containing all the information from a date object and a time object. Like a date object, datetime assumes the current Gregorian calendar extended in both directions; like a time object, datetime assumes there are exactly 3600*24 seconds in every day.

Constructor:

class datetime.datetime(year, month, day[, hour[, minute[, second[, microsecond[, tzinfo]]]]])

The year, month and day arguments are required. tzinfo may be None, or an instance of a tzinfo subclass. The remaining arguments may be ints or longs, in the following ranges:

- MINYEAR <= year <= MAXYEAR
- 1 <= month <= 12
- 1 <= day <= number of days in the given month and year
- 0 <= hour < 24
- 0 <= minute < 60
- 0 <= second < 60
- 0 <= microsecond < 1000000

If an argument outside those ranges is given, **ValueError** is raised.

Class attributes:

- **datetime.min**
- The earliest representable datetime, datetime(MINYEAR, 1, 1, tzinfo=None).
- **datetime.max**
- The latest representable datetime, datetime(MAXYEAR, 12, 31, 23, 59, 59, 999999, tzinfo=None).
- **datetime.resolution**
- The smallest possible difference between non-equal datetime objects, timedelta(microseconds=1).

Instance attributes (read-only):

- **datetime.year**
- Between MINYEAR and MAXYEAR inclusive.
- **datetime.month**
- Between 1 and 12 inclusive.
- **datetime.day**
- Between 1 and the number of days in the given month of the given year.
- **datetime.hour**
- In range(24).
- **datetime.minute**
- In range(60).
- **datetime.second**
- In range(60).
- **datetime.microsecond**
- In range(1000000).
- **datetime.tzinfo**
- The object passed as the tzinfo argument to the datetime constructor, or None if none was passed.