THE ONLY SHIBBOLETH THE WEST HAS IS SCIENCE. IT IS THE PREMISE OF MODERNITY AND IT DEFINES ITSELF AS A RATIONALITY CAPABLE OF, INDEED REQUIRING SEPARATION FROM POLITICS, RELIGION AND REALLY, SOCIETY. MODERNISATION IS TO WORK TOWARDS THIS.

BRUNO LATOUR

THE BOUNDARY BETWEEN SCIENCE FICTION AND SOCIAL REALITY IS AN OPTICAL ILLUSION.

DONNA HARAWAY

INTRODUCTION TO PYTHON

Contents

List of Figures

List of Tables

The longest snake ever held captive is Medusa,

a reticulated python (python reticulatus).

On 12 October 2011, she was measured at

7.67 m long.

Note

This physics text is an OpenSource academic project developed in
abstraction at The Academy. The manuscript is written in LaTeX and
makes use of the `tufte-book` and `tufte-handout` document classes.

http://latex-project.org/ftp.html
https://git-scm.com/downloads

Introduction to OSX Terminal

Intro

Terminal (Terminal.app) is the terminal emulator included in the OS X operating system by Apple. (Wikipedia)

    A terminal emulator is an application that allows access to the operating system. This emulator can be used to perform multiple tasks; it can perform from simple tasks like printing out text, to performing more complex tasks like uploading files to the cloud.

Basic Commands:

```
break : Exit from a For, While, Until or Select loop.
cd : Change Directory
chmod : Change access permissions
clear : Clears terminal screen
curl : Transfer data  from or to a server
date : Display or change the date & time
echo : Display text on screen
exec : Execute a command
false : Do nothing, unsuccessfully
for : Loop command
info : Help info
jobs : List active jobs
reboot : Stop and restart the system
shutdown : Shutdown or restart OS X
sleep : Delay for a specified time
until : Loop command
while : Loop command
!! : Run the last command again
```

Hello World

Here is an example HelloWorld Program written for OS X Terminal.

```
echo Hello World
```

Hello World

Git

Git is a distributed revision control system with an emphasis on speed, data integrity, and support for distributed, non-linear workflows. Git was initially designed and developed by Linus Torvalds for Linux kernel development in 2005, and has since become one of the most widely adopted version control systems for software development.

Please note the below assume you are using a Terminal shell in Linux or OSX operating system. If you are using Windows you will use "dir" instead of "ls" to list files using Command Terminal. Also note the slashes are different for writing file paths. Linux and OSX use forward slash / while Windows uses back slash.

Introduce yourself to Git

```
$ git config --global user.name "Dr Doeg"
$ git config --global user.email doeg@example.com
```

This is the first step you must do when using git for the first time. Tag your commits with Name and Email.

Basic Terminal Commands

Using the terminal you may navigate the file directory. Make, delete, move and rename files and directories.

```
$ cd path/to/project/folder
$ ls
$ cp filename ~/Location/newname
$ mv filename ~/Location/newname
$ rm filename
$ rmdir directoryname
$ touch filename
$ mkdir directoryname
$ nano filename
```

Unix Command	Action
cd	change directory
ls	list files
cp	list files
mv	move files
rm	remove files
rmdir	remove directory
touch	create file
nano	edit file
mkdir	create directory

Table 1: A list of Unix shell commands.

Git Repository

You can get a Git project using two main approaches. The first takes an existing project or directory and imports it into Git. The second clones an existing Git repository from another server.

Setting up a local Repository

```
$ git init
```

Using the command line navigate to the project folder and initialize a git repository.

```
$ git add file2.jpg
$ git add .
```

Add files in the folder to the stage.

Or add all the files.

```
git commit -m "comment on the file changes"
```

Commit the additions.

Push Your Local Repository to GitHub

```
$ git remote add origin https://github.com/<USER>/<REPO>.git
```

Setup the remote repository location on GitHub using your account.

```
$ git remote set-url origin https://..../<USER>/<REPO>.git
```

If you already set up the remote and want to change it use "set-url".

```
$ git push origin master
```

Push the committed structure to the remote server.

Cloning an Existing Repository From GitHub

```
$ cd path/to/whereUwant/folder
```

Navigate to the desired location in file structure.

```
$ git clone https://github.com/<USER>/<REPO>.git
```

Set the location on the GitHub server to place the repositiory.

Working With Branches

Version control is one of the great powers of git.

```
$ git branch
$ git branch branchname
$ git checkout branchname
$ git merge branchname
$ git branch -m newbranchname
$ git branch -D branchname
```

Unix Command	Action
branch	list branches
branch <NAME>	create new branch
checkout <NAME>	switch to new branch
merge <NAME>	merge branch with current
branch -m <NAME>	rename current branch
branch -D <NAME>	delete branch

Table 2: A list of git commands for version control.

Updating an Existing Repository From GitHub

```
$ git fetch
$ git pull -u origin master
```

The sophisticated way to update uses fetch, reviews changes and merges those onto the master branch. The alt the current project folder from the GitHub remote server.

Get Git, Github and More on Git

https://git-scm.com/downloads
https://github.com/
https://git-scm.com/book/en/v2

Download git and register an account at GitHub. Look at the official documentation for more information.

Git

What the git is

Github is a website that can make people able to share their ideas. People can upload and download by pushing and cloning in order. When the git saves the files, git gets the checksum with sha-1 hash before it saves. Therefore, it is difficult to lose and check the files. This checksum is a 40-character string composed of hexadecimal characters and calculated based on the contents of a file or directory structure in Git.

Getting a git repository

There are two main approaches getting a git repository. The first one take an existing project or directory it into git. The second one clone an existing git repository from another server.

If you want to move an existing project in git, you need to go to the project's directory and type:

-git init
-git add .
-git commit -m 'comment on the file changes'

If you want to clone a git progect, you need to do so like this:
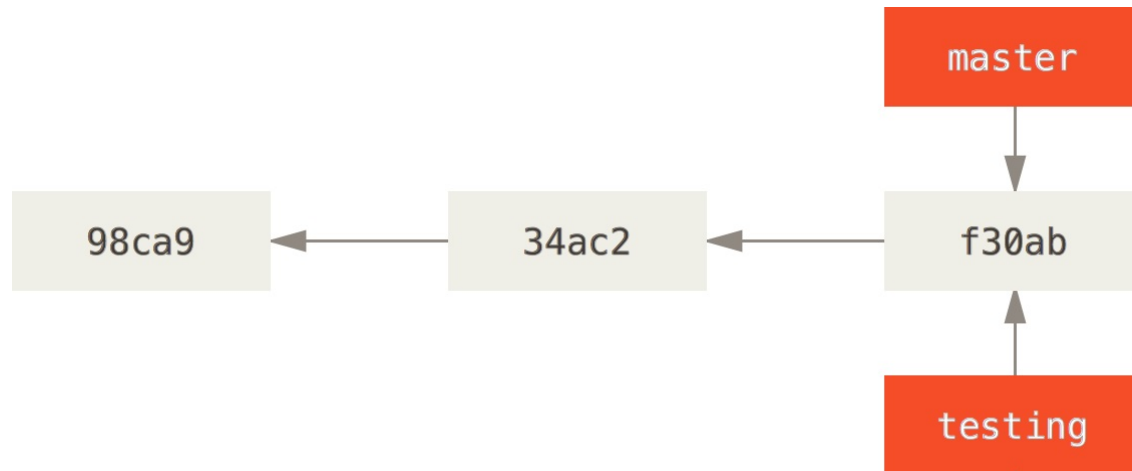
-git clone https://github.com/YOUR-USERNAME/YOUR-REPOSITORY

Branch

The git branch commend is a branch administration tool. This commend can show the list of all branch, create a new branch , delete a branch and rename a branch

If you want to create a new branch, The git branch commend can make a testing branch:

- git branch testing

Git is differant the other virsion administration tool. Git has a special point called HEAD. This is a pointer to local branch you'r currently on. In this case, your git is still on master branch. The git branch commend makes a new branch, dosen't move the branch.

Merge

The git merge commend merge other branch init the branch you have checked out. If you use the "git merge <branch>" commend, the branch would merge.

what is latex

LaTeX is a document preparation system for high-quality typesetting. It is most often used for medium-to-large technical or scientific documents but it can be used for almost any form of publishing.(Basically its a high-tech version of a pdf document maker that can do much more stuff than normal word documents.)
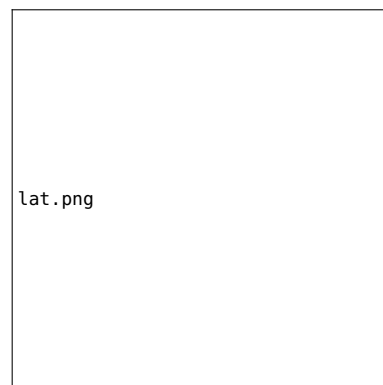
```
lat.png
```

Figure 1: This is the logo of latex.

this is how you start a latex file.

```
\documentclass{tufte-handout}

\title{Latex}

\author[The Academy]{Tony/Zekang Lin}

\begin{document}
```

Here you start you file by telling what kind of file you are making and the title and author of it. And begin the document.

And begin the document. remenber to have enddocument at the end of the paper.

this is some useful things you can do in latex file.

```
\begin{marginfigure}%
\includegraphics[width=\linewidth]{XXXXX.png}
\caption{This is the logo of latex.}
\label{fig:marginfig}
\end{marginfigure}


\marginnote[30pt]{DXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX}


\begin{shaded}
\begin{verbatim}
```

you can use margin figure or figure to include pictures as a note or just something that you want to show.it will auto matically lable the picture as figure + the number of the image.

use marginnote to add notes.

shaded can help you shade what you are going to write and verbatim will allow you to write your codes in latex.

why do we use latex.

Latex is easy to use and there are many stuff that latex will
automaticallydo for you, such as it will automatically write
the date that you last edited and autometically lable the number
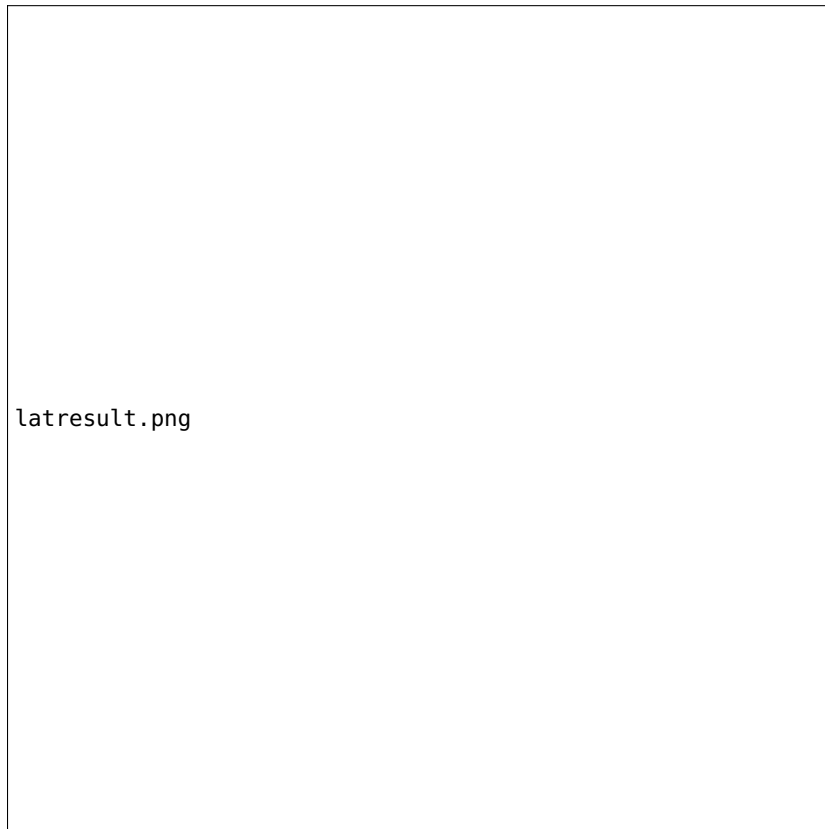of images you added to the paper.

this is what a Latex paper can look like:

latresult.png

Figure 2: this is what latex paper can look like.

Array 1-D, 2-D, 3-D

Intro

Array is like a storage, it can fill with string or integer. In 1-D, 2-D it can also represents the x-axis and y axis.

Creating arrays

Arrays is created buy blanket.

Example:

```
a=[ ]        *a is the array name that you want.
```

The things in the [ ] and be store and when you want to access it you will need its position in the array and type like a[0]

Example:

```
a=["apple","orange","banana"]
```

If you want to print banana form the array, you may want to type

```
print a[2]
```

Filling arrays

Everything can be store in the array, strings, integers, arrays. When you create an array you can fill things in it as the default things that the array have.

Example:

```
a=["Billy","Bud",90,60,50]
b=["Anne","Chow",90,95,100]
c=["Jen","Bo",60,80,90]
```

If you want to add things into the array that u create already, you can use

```
array_name=array_name+[Things you want to add]
```

Example:

```
a=[apple]
```

and now I want to add orange into it, so we add

```
a=a+[orange]
```

To create 2-D or more array we need to create array in the nested for-loop.

Example:

```
a=[]
for i in range(N): *N how long you want the array to be
    b=[]    *This is a temporary array to generate every array inside the main array.
    for j in range(N):
        *Things you want to put in the array by b=b+[ ]
    a=a+[b]    *Here put the temporary array back to the main array.
```

Traversing array

Traversing array is visiting each element in the array and do something. In 1-D we can do it with for loop to identify things in array.
Example:

```
a=[1,2,3]
for i in range(len(a))    *len(a) =  Numbers of elements in the array
    *Things put here can edit the specific element a[i]
```

In 2-D we start using nested for-loop to identify the x-axis and y-axis. So we use nested for-loop to traversing it too.

Example:

```
a=[[0,1],[0,0],[0,1]]
for i in range(len(a)):
    for j in range(len(a)):
        *Things put here can edit the specific element a[i][j]
```

In 3-D we use more for-loop to identify the more dimension.

Example:

```
a=[[[0,0],[0,0]],[[0,0],[0,0]]]
for x in range(len(a)):
    for y in range(len(a)):
        for z in range(len(a)):
            *Things put here can edit the specific element a[x][y][z]
```

Integers and Floating Point Numbers

What is Integer?

An integer (from the Latin integer meaning "whole") [note 1] is a number that can be written without a fractional component. For example, 21, 4, 0, and âĹŠ2048 are integers, while 9.75, 5Â¡, and âĹŽ2 are not.
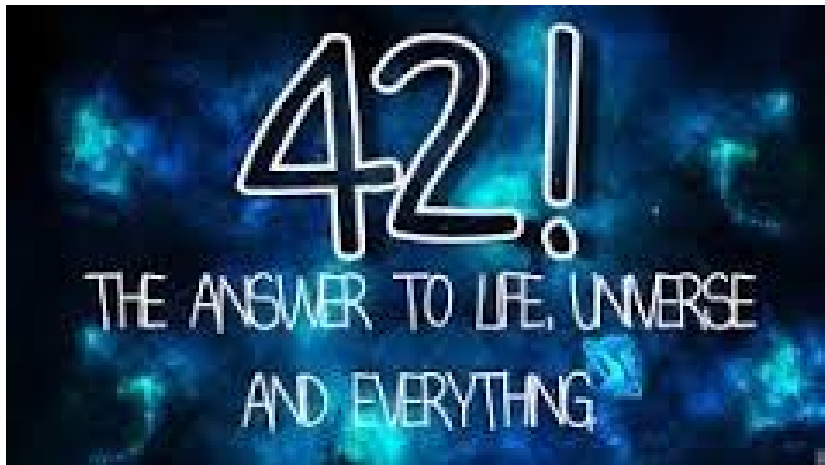

[?]

Figure 3: The ultimate answer of the universe is also an Integer!

Floating point!

Also called floats, they represent real numbers and are written with a decimal point dividing the integer and fractional parts. You can convert an integer to a floating number by your python and vice versa! :

```
Type int(x) to convert x to a plain integer.
```

```
Type: int(4.22222)
Result: 4
```

```
Type float(x) to convert x to a floating-point number.
```

```
type: float(1)
Result: 1.0
```

Number Type Conversion

Python converts numbers internally in an expression containing
mixed types to a common type for evaluation. But sometimes, you
need to coerce a number explicitly from one type to another to satisfy
the requirements of an operator or function parameter.

Definition:

Booleans:

In Python we have the following terms (characters and phrases) for determining if something is "True" or "False." Logic on a computer is all about seeing if some combination of these characters and some variables is True at that point in the program.

and
or
not
!= (not equal)
== (equal)
>= (greater-than-equal)
<= (less-than-equal)
True
False
We use these characters to make the truth or not.
NOT:
not False = True
not True = False
OR:
True or False = True
True or True = True
False or True = True
False or False = False
AND:
True and False = False
True and True = True
False and True = False
False and False = False
NOT OR:
not (True or False) = False
not (True or True) = False
not (False or True) = False
not (False or False) = True
NOT AND:
not (True and False) = True
not (True and True) = False
not (False and True) = True
not (False and False) = True
!=:
1 != 0 = True
1 != 1 = False
0 != 1 = True

```
0 != 0 = False
==:
1 == 0 = False
1 == 1 = True
0 == 1 = False
0 == 0 = True
```

While Loops

A while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.

The condition may be any expression, and true is any non-zero value. The loop iterates while the condition is true.When the condition becomes false, program control passes to the line immediately following the loop.In Python, all the statements indented by the same number of character spaces after a programming construct are considered to be part of a single block of code. Python uses indentation as its method of grouping statements.

Figure 4: Flow diagram about how the while loop works

Python code example

```
count = 0
while (count < 9):
    print 'The count is:', count
    count = count + 1

print "Good bye!"
```

Output

```
>>>
The count is: 0
The count is: 1
The count is: 2
The count is: 3
```
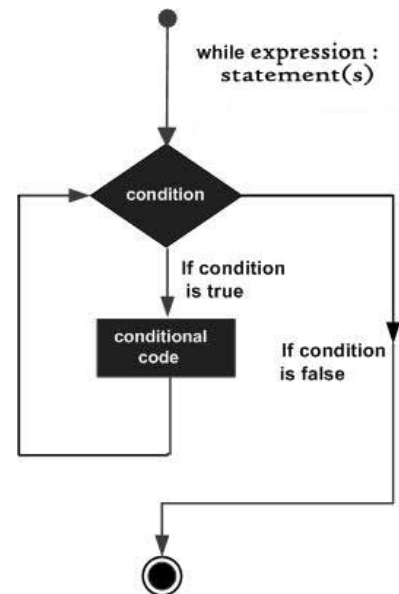
The code will produce the following output.

```
The count is: 4
The count is: 5
The count is: 6
The count is: 7
The count is: 8
Good bye!
>>>
```

Infinite Loop

A loop becomes infinite loop if a condition never becomes **FALSE**. You must use caution when using while loops because of the possibility that this condition never resolves to a FALSE value. This results in a loop that never ends. Such a loop is called an infinite loop.

An infinite loop might be useful in client/server programming where the server needs to run continuously so that client programs can communicate with it as and when required.

```
var = 1
while var == 1:
    num = raw_input("Enter a number  :")
    print "You entered: ", num


print "Good bye!"
```

This python code is an example of how infinite loop can be created.

```
Enter a number  :X
You entered:  x
Enter a number  :Y
You entered:  Y
Enter a number  :Z
You entered:  Z
Enter a number between :
```

This code creates an infinite loop where it will need your input of any number.Once you input any number, it will output it like if you input "X" it will show back "X".

To break the loop you will either need to add the "break" command in your code OR press **CTRL+C** to exit the program.

Using else statements with while loops

Python supports to have an else statement associated with a loop statement.

If the else statement is used with a while loop, the else statement is executed when the condition becomes false.

```
count = 0
while count < 5:
   print count, " is  less than 5"
   count = count + 1
else:
   print count, " is not less than 5"
```

The following code illustrates the combination of an else statement with a while statement that prints a number as long as it is less than 5, otherwise else statement gets executed.

```
>>>
0 is less than 5
1 is less than 5
2 is less than 5
3 is less than 5
4 is less than 5
5 is not less than 5
>>>
```

This is the output in python when the code above is executed.

While Loops

A while loop is a function, which needs a boolean statement to run, in order to prit out a list of results. The while loop will print out as many resultts as possible, until the boolean statement stops being true. For example:

```
i=0
while (i<5):
    print i
    i=i+1
```

Now, the boolean statement inserted here is "i<5", meaning that "i" must be less than 5. The next function now commands the system to print "i". The result would be:

```
0
1
2
3
4
```

This is because after the computer was demanded to print out "i", the function "i==i+1", was entered, meaning that it should print out all the numbers that make the bolean statement rue, until the number is five, making it false.

There are also many ways you could manipulate this code. A break may be inserted as follows:

```
i=0
while (i<5):
    print i
    i=i+
    if i==4:
        break
```

The result would be:

```
0
1
2
3
```

Excluding "4", because as a result of the break, the computer is now told not to print anything from the number 4.

However, if the break came before "i=i+1" like:

```
i=0
while (i<5):
    print i
    if i==4:
        break
    i=i+1
```

The result would be:

```
0
1
2
3
4
```

Bases

The word "base" in mathematics is used to refer to a particular mathematical object that is used as a building block. The most common uses are the related concepts of the number system whose digits are used to represent numbers. In common, people use decimal system. However, the computers are compsed with binary system. Also, SHA-256 gives the hexadecimal code.

Binary

Binary is composed with only 0 and 1. Each of digits represents 2 to the power of something. From the right to left, it starts from 2^0 to increasing the power.
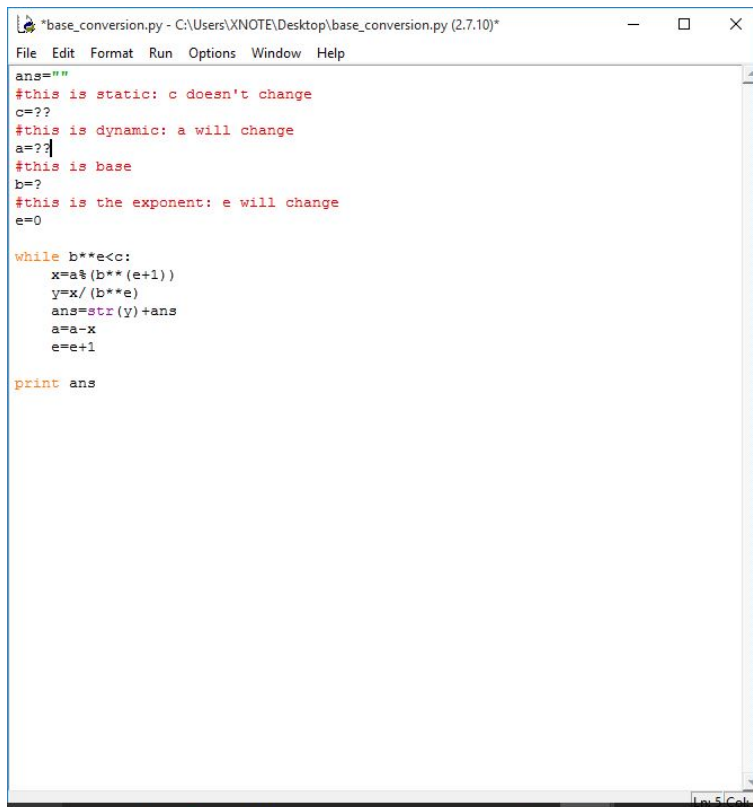
Decimals

Decimal system is what people use in usual days. It is composed with 0,1,2,3,4,5,6,7,8, and 9, and it starts from 10^0 and increases the power of 10 when it goes the next digit. The reason why people use the decimal system is that the human has 10 fingers, so the calculation is easier and more simple.

Hexadecimals

Hexadecimal system is used for computer languages such as C language/C++ and SHA-256.It is composed with 1,2,3,4,5,6,7,8,9,a,b,c,d,e, and f. When the decimal numbers are hashed by SHA-256, it is more difficult if hashed number starts with 0 and have many 0s in the number. The number in front of x is the number of 0s which is leading.

Conversion

It is the sample code of the base conversion. ? is what the base is, so it can be any decimal number. ?? can also be the any number,

```
ans=""
#this is static: c doesn't change
c=??
#this is dynamic: a will change
a=??
#this is base
b=?
#this is the exponent: e will change
e=0

while b**e<c:
    x=a%(b**(e+1))
    y=x/(b**e)
    ans=str(y)+ans
    a=a-x
    e=e+1

print ans
```

but decimal numbers. It changes from decimal numbers to base of ? numbers. a needs to be subtracted by x for the next step getting ? to the power of next exponent. Then, repeat the previous steps with e added 1. And then, repeat for next step again. At the last, however, y whicch is the quotient of the last x divided b to the power of e should be concantenated with ans. The quotient becomes the first number of the ? base numbers.

Array 1-D, 2-D, 3-D

Intro

Array is like a storage, it can fill with string or integer. In 1-D, 2-D it can also represents the x-axis and y axis.

Creating arrays

Arrays is created buy blanket.

Example:

```
a=[ ]        *a is the array name that you want.
```

The things in the [ ] and be store and when you want to access it you will need its position in the array and type like a[0]

Example:

```
a=["apple","orange","banana"]
```

If you want to print banana form the array, you may want to type

```
print a[2]
```

Filling arrays

Everything can be store in the array, strings, integers, arrays. When you create an array you can fill things in it as the default things that the array have.

Example:

```
a=["Billy","Bud",90,60,50]
b=["Anne","Chow",90,95,100]
c=["Jen","Bo",60,80,90]
```

If you want to add things into the array that u create already, you
can use

```
array_name=array_name+[Things you want to add]
```

Example:

```
a=[apple]
```

and now I want to add orange into it, so we add

```
a=a+[orange]
```

To create 2-D or more array we need to create array in the nested
for-loop.

Example:

```
a=[]
for i in range(N): *N how long you want the array to be
    b=[]    *This is a temporary array to generate every array inside the main array.
    for j in range(N):
        *Things you want to put in the array by b=b+[ ]
    a=a+[b]    *Here put the temporary array back to the main array.
```

Traversing array

Traversing array is visiting each element in the array and do some-
thing. In 1-D we can do it with for loop to identify things in array.
Example:

```
a=[1,2,3]
for i in range(len(a))    *len(a) =  Numbers of elements in the array
    *Things put here can edit the specific element a[i]
```

In 2-D we start using nested for-loop to identify the x-axis and y-axis.
So we use nested for-loop to traversing it too.

Example:

```
a=[[0,1],[0,0],[0,1]]
for i in range(len(a)):
    for j in range(len(a)):
        *Things put here can edit the specific element a[i][j]
```

In 3-D we use more for-loop to identify the more dimension.

Example:

```
a=[[[0,0],[0,0]],[[0,0],[0,0]]]
for x in range(len(a)):
    for y in range(len(a)):
        for z in range(len(a)):
            *Things put here can edit the specific element a[x][y][z]
```

Arrays

In a nutshell, arrays are lists of data. Most commonly, they are sets
of integers, although they can also be sets of strings. There are many
uses for an array, such as being used as a database to count occur-
rences. The simplest arrays are one-dimensional arrays, which look
like this.

```
[0,1,2,3,4]
["a","b","c","d","e"]
```

These arrays can be described with several different characteristics.

1. They are both an array of length 5.

2. They are both one dimensional arrays (trust me, this will make
   more sense).

3. The first is a list of integers, and the second is a list of strings.

Array generation

Arrays are generated in a program by first creating an empty array,
or by creating an array that is already filled with whatever you'd like.

```
a = []
savoryfillings = ["meat","cheese","potato"]
```

Adding to an array

Adding to an array, also known as "concatenation", is also a very
simple process, much like generating an array. A simple way to fill an
array with, let's say, zeroes, is with a for loop.

```
a = []
for i in range(5):
a = a + [0]
print a
#this program will yield [0, 0, 0, 0, 0]
```

Note how the zero is in brackets. This will signify to the inter-
preter that this zero is intended to be a unit of the array.

Strings can also be concatenated to an array of strings.

```
>>> savoryfillings = ["meat","cheese","potato"]
>>> savoryfillings = savoryfillings + ["spinach"]
>>> print savoryfillings
['meat', 'cheese', 'potato', 'spinach']
```

Referencing certain parts of an array

Referencing certain parts of an array is rather simple. A simple pro-
gram that does such thing and prints the referenced points is dis-
played below. Remember that Python starts counting at 0, not 1!

```
>>> a = [1, 2, 3, 4]
>>> x = a[0]
>>> y = a[1]
>>> #referencing points of array is done by (nameofarray)[pointinarray]
>>> print x
>>> print y
1
2
```

The command 'len'

The 'len' command takes the length of a given array and converts it
into an integer.

```
>>> a = [1, 2, 3]
>>> x = len(a)
>>> print x
3
```

'len' can also be used to do something to every point of an array,
with the use of a for loop.

```
>>> a = [0, 1, 2, 3, 4, 5]
>>> for i in range(len(a)):
a[i] = a[i] + 1

>>> print a
[1, 2, 3, 4, 5, 6]
```

Two-/three-dimensional arrays

Multidimensional arrays can be created through the use of nested
for loops - for example, the program below generates a 5x5 array of
zeroes.

```
>>> a = []
>>> for i in range(5):
b = []
for j in range(5):
b = b+[0]
a = a+[b]
>>> print a
[[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]
```

Note how this two-dimensional array has five arrays within it,
each with five zeroes.

Application of arrays

Arrays can be used to catalog data. This program logs

```
import random
mastercount = []
for i in range (10):
    mastercount = mastercount + [0]
for i in range (1000):
    x = random.randint(0,9)
    mastercount[x] = mastercount[x]+1
print mastercount
#sample yield of program - [95, 103, 96, 98, 107, 108, 101, 107, 102, 83]
```

For Loops, Nested For Loops

For in loop has the ability to iterate over the items of any sequence, such as a list or a string.

If a sequence contains an expression list, it is evaluated first. Then, the first item in the sequence is assigned to the iterating variable iterating var. Next, the statements block is executed. Each item in the list is assigned to iterating var, and the statement(s) block is executed until the entire sequence is exhausted.

PYTHON CODE

Syntax:

```
for var in sequence:
    statements
```

Example 1

```
for i in ["hello", "hey", "yo"]:
    print i
```

The goal is to print all the elements in the array

Output 1

```
>>> Hello
    hey
    yo
```

Example 2

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

The goal is to print all the number in range 5. Note, that Python starts counting from 0

Output 2

```
>>> 1
    2
    3
    4
    5
```

Nested for loops

```
Python programming language allows to use one loop inside
another loop. Following section shows few examples to illustrate
the concept.
```

Syntax

```
for iterating_var in sequence:
   for iterating_var in sequence:
      statements(s)
   statements(s)
```

Example 1

```
for i in range(5): #loop everything indented 5 times
    for j in range(3): #loop everything indented 3 times
        print i #print output of i
```

The goal is to print every number from 0 to 5 three times in a row.

Output 1

```
>>> 0
    0
    0
    1
    1
    1
    2
    2
    2
    3
    3
    3
    4
    4
    4
```

Example 2

```
import random #importing "random" library that will take random numbers for our program
x = [] #creating initial, empty array that will be fulfilled with other arrays later
for i in range(5): #loop everything indented 5 times
    r=[] #creates 5 more arrays
    for j in range(5): #loop everything indented 5 times
        if random.uniform(0,10)<3: #compare if randomly chosen number is smaller than 3
            r=r+[1] #if it is, add value of "1" inside the secondary array - r
        else: #or
            r=r+[0] #if it is bigger, add value of "0" inside the secondary array - r
    x.append(r) #add all of these secondary arrays to our main - x array.
    #Basically we create 2D array.
print x #print our 2D array to see the output.
```

The goal is to create a 2D arrays that would be fulfilled with values of "1" or "0" depending on which number was randomly chosen by the computer

Output 2

```
>>> [[0, 0, 0, 1, 1], [1, 0, 0, 0, 0], [0, 0, 0, 0, 0],
    [0, 1, 0, 0, 0], [0, 0, 0, 0, 0]]
```

However, the numbers that you see here isn't the only output you can get. We used random uniforms and each time it will give different values.

Libraries

Introduction

Libraries in Python are extensions to the basic Pyhton coding. Python comes with some libraries of its own. But it is also possible to write your own libraries. But before you can use libraries you have to import the libraries that you want to use in your script. There are several ways how you can import libraries. Libraries are always imported at the beginning of a scrip.

Importing Python Libraries

The easiest way to import libraries is to use the import function. For these examples we will use the random library.

```python
import random
print random.uniform(1,10)
```

With this you have to put the libraries name in front of the function of the library

If you want to rename a library before you are using it you can do the following

```python
import random as rndm
print rndm.uniform(1,10)
```

If you dont want to have to write a library name in front of it at all you can do

```python
from random import *
print uniform(1,10)
```

There is one other option how you can import libraries. If you use all of what we learned before we can use the following.

```python
from random import uniform as makeRandom
print makeRandom(1,10)
```

With this you can import a single function of a libary and you name the function.

Importing custom libraries

If you want to use libraries you or someone else has written in
python you can do that also. First you have to make sure that the
script you want to import is in the same folder as the script you
want to import it into. Lets asume we have to following script we
want to use as a libary.

```python
def Bla():
    print "Bla"

def MyFunction(A)
    print A+A
```

Lets assume the scripts name is lib. Now if we want to use this in our
main script we can do eaither of our ways. Use the file name as the
libraries name.

```python
import lib
import lib as mylib
from lib import Bla as Tell
```

Random

Intro

Random is a library that you can used to generate random numbers
in python.

Creating random numbers

You can use a code to generate random numbers.

```
import random     *You need to import random before you generate random numbers
random.uniform(minimum,maximum)
```

while minimum and maximum number can be intergers(0) or floats(0.0)

Usage of random number

Simply here is a example of how to add random numbers to an array

```
import random
a=[]
for i in range(N):     *N is the the number of how many random numbers you want to add to the array
    a=a+[random.uniform(0,100)]
        *0 is the minimum number and 100 is the maximum number, they are variable, you can change to
         numbers you want.
```

Here is a example of using random function to randomly fill a 2-D
array with 0 and 1 by how many percentage you want it is filled.

```
import random
def fill(a,p):    *a is the name or you 2-D array    *p is the percentage you want, such as: 0.1, 0.25....
    for i in range(len(a)):     *len(a) is the length of you array
        for j in range(len(a)):
            if random.uniform(0,1)<p:
        a[i][j]=1
    else:
        a[i][j]=0
```

Statistics Library

Introduction

This statistics library includes eight functions that we can use to deal with a set of data.

Tips

- *Additional libraries are needed*

- *Multiple functions are needed for some operations*

Zero

```
def zeros(n):
    a=[]
    for i in range(n):
        a=a+[0]
    return a
```

(1)

This function can be used to create a consecutive and repeating array(all elements are the same).

Example:

```
def zero(n):
    a=[]
    for i in range(n):
        a=a+[0]
    return a

print zero(9)
```

Output: [0, 0, 0, 0, 0, 0, 0, 0, 0]

Summing an array

```python
def sum_array(a):
    s=0
    for i in range(len(a)):
        s=s+a[i]
    return s
```

$$(2)$$

This function can help us to sum up all the elements in an array.

Example:

```python
a=[1,2,3,4]
def sum_array(a):
    s=0
    for i in range(len(a)):
        s=s+a[i]
    return s
print sum_array(a)
```

Output: 10

Finding means

```python
def avg(a):
    return sum_array(a)/len(a)
```

$$(3)$$

This function can help us to calculate the mean of all the data in an array, and we need assistance of the sum function.

Example:

```python
a=[23,19,25,21]
def sum_array(a):
    s=0
    for i in range(len(a)):
        s=s+a[i]
    return s

def avg(a):
    return sum_array(a)/len(a)

print avg(a)
```

Output: 22

Variance

```
def var(a):
    s=0
    for i in range(len(a)):
        s=s+a[i]**2
    m=avg(a)
    return (s/len(a)-m**2)
```

$$(4)$$

This function can help us to find the variance of the data.Also, we need the assistance of the average function.

Example:

```
a=[23,19,25,21]
def sum_array(a):
    s=0
    for i in range(len(a)):
        s=s+a[i]
    return s

def avg(a):
    return sum_array(a)/len(a)

print avg(a)

def var(a):
    s=0
    for i in range(len(a)):
        s=s+a[i]**2
    m=avg(a)
    return (s/len(a)-m**2)
print var(a)
```

Output: 5

Construct an array of random numbers

```
def rand_array(n,mini,maxi):
    a=[]
    for i in range(n):
        a=a+[random.uniform(mini,maxi)]
    return a
```

$$(5)$$

We can use this function to construct an array filled with random numbers. We need the random library to run the function.Here n represents the number of elements in the array; mini is the minimum value of the elements;maxi is the maximum value

To fill a histogram

Library "graphics" is needed*

```
def histogram(mini,maxi,bins,a):
    h=zeros(bins)
    w=(maxi-mini)/bins
    for i in range(len(a)):
        for j in range(bins):
            if (a[i]>(mini+j*w))and a[i]<(mini+(j+1)*w):
                h[j]=h[j]+1
    return h
```

$$(6)$$

The four arguments in parenthesis are decisive for the histogram. Here mini represents the minimum value in the data; maxi represents the maximum value in the data; bin represents the number of different groups of data; a is the number of all the data.

Find the maximum value

```
def maximum(a):
    m=0
    for i in range(len(a)):
        if a[i]>m:
            m=a[i]
    return m
```

$$(7)$$

This funciton can help us to identify the maximum value in the data.

Drawing a histogram

Library "graphics" is needed
To draw a histogram, we first need to define a function that can draw a bargragh:

```
def bar(a,win):
    win.setCoords(-1,-1,len(a)+1,maximum(a)+1)
    bl=[]
    tr=[]
    rec=[]
    for i in range(len(a)):
        bl=bl+[Point(i,0)]
        tr=tr+[Point(i+a,a[i])]
        rec=rec+[Rectangle(bl[i],tr[i])]
        rec[i].draw(win)
```

(8)

Then we combine the "bar" function and the "histogram" function.
Here is just an example:

```
def main():
    win=GraphWin()
    a=zeros(400)
    for i in range(len(a)):
        a[i]=sum_array(rand_array(10,0,1))
    histo=histogram(0.,10.,7,a)
    bar(histo,win)
main()
```
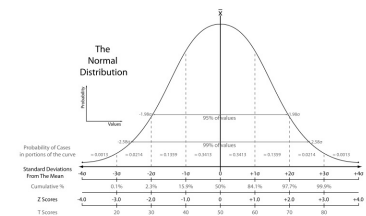
(9)

Statistics

Figure 5: More probability density is found as one gets closer to the expected (mean) value in a normal distribution. Statistics used in standardized testing assessment are shown. The scales include standard deviations, cumulative percentages, percentile equivalents, Z-scores, T-scores, standard nines, and percentages in standard nines..

Introduction

Statistics is the study of the collection, analysis, interpretation, presentation, and organization of data.[1] In applying statistics to, e.g., a scientific, industrial, or societal problem, it is conventional to begin with a statistical population or a statistical model process to be studied. Populations can be diverse topics such as "all persons living in a country" or "every atom composing a crystal". Statistics deals with all aspects of data including the planning of data collection in terms of the design of surveys and experiments.

Statistics library

```
import random

def zeros(n):
a=[]
for i in range(n):
a=a+[0]
return a
def sum_array(a):
s=0
for i in range(len(a)):
s=s+a[i]
return s
def rand_array(n,mini,maxi):
a=[]
for i in range(n):
a=a+[random.uniform(mini,maxi)]
return a
```

In the begin, I need type in "import random" to start this statistics code .

The first code in statistics library is the explanation of the zero, it is a basic mode to calculate.

The second code is to define the sum of array, it will help user to calculate the sum when the array is longer than affordability.

The third code is to define the rand of array, this code is start more complex than before, we need the n, mini and maxi to finish the calculation

```
def avg(a):
return sum_array(a)/len(a)
def var(a):
s=0
for i in range(len(a)):
s=s+[a[i]**2]
m=avg(a)
return (s/len(a)-m**2)
def maximum(a):
m=-math.inf
for i in range(len(a)):
if a[i]>m:
m=a[i]
return m
```

The fourth code is to know the average of the array, we will use the sum of array devide by lenth of array to calculate this, and we already know how to type the sum of array, so it is not a no clue function.

In the fifth code, the var is when you define a function parameter and it is a variable number of parameters of meaning, so we can use this code to figure out it very convenient.

The sixth code is to calculate the maximum.

```
def histogram(mini,maxi,bins,a):
h=zeros(bins)
w=(maxi-mini)/bins
for i in range(len(a)):
for j in range(bins):
if a[i]>(mini+j*w) and a[i]<(mini+(j+1)*w):
h[j]=j[j]+1
return h
def bargraph(a):
win=graphWin("BarGraph",400,400)
win.SetCoords(-1,-1,len(a)+1,maximum(a)+1)
for i in range(len(a)):
rec=Rectangle(Point(i,0),Point(i+1,a[i]))
rec.draw(win)
def main():
gauss=[]
for i in range(100):
gauss=gauss+[sum_array(rand_array(10,0,1))]
bargraph(histogram(0,10,10,gauss))
```

The seventh code is the histogram, we need use the value of mini,maxi,bins and a.

The last two code is bargraph and main, be honest, those two code is relatively strange for me, but in the statistics library, it will be a good supporting code to reduce your time for anytime you need to use those code.

Graphics Library

The graphics library is a library for making easy graphical objects in python. It was written by John Zelle for use with his book Python Programming: An Introduction to Computer Science. This chapter is going to discuss the topics we covered during class. To use the library you have to import it first. For an explanation on how to import libraries see the section of the book.

Creating a basic window

The graphics library works as a programm of it own. A window gets updated as long as it gets changed or is waiting for a new command. To draw somthing on a window we first have to create a window. A creation follows this pattern:

```
1  Window = GraphWin(WindowName, Width, Height)
2  Window.setCoords(StartX,StartY,WidthPixel,HeightPixel)
```

Drawing on windows

Now that we know how to create a window we can start drawing on it. There are various objects we can draw on windows. Because the graphics library uses its own coordinates system we need to use it for creating objects.

```
1  Point(2,4) #Creates a point at 2,4
```

To actually draw in windows we have to use the draw command.

```
1  Window = GraphWin(WindowName, Width, Height)
2  Window.setCoords(StartX,StartY,WidthPixel,HeightPixel)
3  Obj = OurObject
4  Obj.draw(Window)
```

In this example we are drawing Obj to our window.

Rectangles

In python rectangles are drawn from the down left point to the top right point.

```
1  Rect = Rectangle(Point(X1,Y1),Point(X2,Y2))
```

This creates a rectangle calles Rect at the position X1 and Y1 with the width X2 and height Y2. After you have created your basic reactangle you can also color it.

```
1  Rect = Rectangle(Point(X1,Y1),Point(X2,Y2))
2  Rect.setFill(color_rgb(255,0,0)) #Set the color for the reactangle
```

Circles

Of course you can also add circles to your window. Circles are drawn from their middle point to the setting of the size of their diameter.

```
1  C = Circle(Point(X,Y),Point(RX,RY))
```

The Point(X,Y) is defining the position of the circle. The Point(RX,RY) is defining the radius of the circle.

Text labels

On a window it is very important to be able to use text. Because of that the graphics library also comes with a text object for windows.

```
1  T = Text(Point(X,Y),StringText)
```

The example is self explaining. But this text is nothing special. To make it cool we need to modifie it a bit.

```
1  Text.setFace(Font)
2   #Availible Fonts are 'helvetica','arial','courier','times roman'
3  Text.setSize(Size) #Size of the text
4  Text.setStyle(Style)
5   #Availible styles 'bold','normal','italic', 'bold italic'
6  Text.setTextColor(Color) # Use color_rgb(r,g,b) as a color
```

Other usefull methods

There are some other usefull methods for the graphics library. Obj
can be any object of the graphics library.

```
Obj.setFill(Color) #Sets the color
Obj.clone() #Creates a new object with the same properties as Obj.
Obj.move(X,Y) #Moves the object
color_rgb(r,g,b) #Makes a color for use with this library
Window.getMouse() #Stops the programm until you press on the screen
```

Example code

This is an example code of the graphics library.

```
from graphics import *

Win = GraphWin("Example Window",400,400)
Win.setCoords(0,0,50,50)
T = Text(Point(25,40), "Text")
T.draw(Win)
T.setFill(color_rgb(0,100,100))
T.setStyle("bold")
T.setFace("arial")
T.setSize(30)

Rect = Rectangle(Point(20,20),Point(30,30))
Rect.draw(Win)
Rect.setFill(color_rgb(255,0,0))

for i in range(10):
    C = Circle(Point(15+i*2,i),i)
    C.draw(Win)
    C.setFill(color_rgb(20*i,18*i,7*i))
```
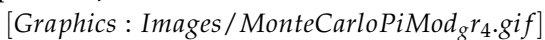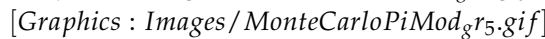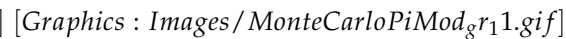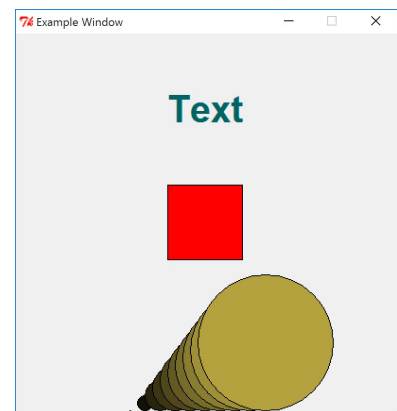
Monte Carlo

Monte Carlo simulations are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. They are often used in physical and mathematical problems and are most useful when it is difficult or impossible to use other mathematical methods. Monte Carlo methods are mainly used in three distinct problem classes:[1] optimization, numerical integration, and generating draws from a probability distribution. In Class we have used the Monte Carlo simulation for the probability function.

We have estimated the value of pi

We start the familiar example of finding the area of a circle. Figure 1 below shows a circle with radius r=1 inscribed within a square. The area of the circle is $Pi * r^2 = Pi * 1 = Pi$ and the area of the square is 4 The ratio of the area of the circle to the area of the square is $[Graphics : Images/MonteCarloPiMod_gr_4.gif]$

$[Graphics : Images/MonteCarloPiMod_gr_5.gif]$ $[Graphics : Images/MonteCarloPiMod_gr_11.gif]$

```
import random
x=random.uniform(-1,1)
y=random.uniform(-1,1)
n=0.
#n is the number of random points in the circle
p=999999
#p is the number of random prints
for i in range(p):
    x=random.uniform(-1,1)
    y=random.uniform(-1,1)
    if (x*x+y*y)<1:
     n=n+1
pi=4*(n/p)
print pi
```

As you increase P the estimation will be more accurate. for p=999999

```
>>>
3.14152314152
>>>
```

Whereas for p=100

```
>>>
3.24
>>>
```

Game of Life

The Game of Life, also known simply as Life, is a cellular automaton devised by the British mathematician John Horton Conway in 1970. The "game" is a zero-player game, meaning that its evolution is determined by its initial state, requiring no further input. One interacts with the Game of Life by creating an initial configuration and observing how it evolves.

Game of Life Rules

The universe of the Game of Life is an infinite two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, alive or dead. Every cell interacts with its eight neighbours, which are the cells that are horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

First
   Any live cell with fewer than two live neighbours dies, as if caused by under-population.

Second
   Any live cell with two or three live neighbours lives on to the next generation.

Third
   Any live cell with more than three live neighbours dies, as if by over-population.

Fourth
   Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

Here is the code we wrote in class:

```
[frame=single]

import random
from graphics import *
```

```python
#this function creates an NxN array filled with zeros
def empty(N):
    a=[]
    for i in range(N):
        b=[]
        for j in range(N):
            b=b+[0]
        a=a+[b]
    return a


#this function fills the array a with a portion p of live cells
def fill(a,p):
    N=len(a)
    for i in range(N):
        for j in range(N):
            if random.uniform(0,1)<p:
                a[i][j]=1


def update(A,B):
    N=len(A)
    for i in range(N):
        for j in range(N):
            neigh=A[(i-1)%N][(j-1)%N]+A[(i-1)%N][j]+A[(i-1)%N][(j+1)%N]+A[i][(j-1)%N]+A[i][(j+1)%N]+
            A[(i+1)%N][(j-1)%N]+A[(i+1)%N][j]+A[(i+1)%N][(j+1)%N]
            if A[i][j]==0:
                if neigh==3:
                    B[i][j]=1
                else:
                    B[i][j]=0
            else:
                if neigh==2 or neigh==3:
                    B[i][j]=1
                else:
                    B[i][j]=0



def gen2Dgraphic(N):
    a=[]
    for i in range(N):
        b=[]
        for j in range(N):
            b=b+[Circle(Point(i,j),.49)]
        a=a+[b]
```

```
    return a

def push(B,A):
    N=len(A)
    for i in range(N):
        for j in range(N):
            A[i][j]=B[i][j]


def drawArray(A,a,window):
#A is the array of 0,1 values representing the state of the game
#a is an array of Circle objects
#window is the GraphWin in which we will draw the circles
    N=len(A)
    for i in range(N):
        for j in range(N):
            if A[i][j]==1:
                a[i][j].undraw()
                a[i][j].draw(window)
            if A[i][j]==0:
                a[i][j].undraw()

N=10             #N is the number of live cells you start with
win = GraphWin("title",400,400)
win.setCoords(-1,-1,N+1,N+1)
grid=empty(N)
grid2=empty(N)
circles=gen2Dgraphic(N)
fill(grid,0.3)

while True:
    drawArray(grid,circles,win)
    update(grid,grid2)
    push(grid2,grid)
```