HOW TO CODE LIKE SHERLOCK HOLMES: The Science Of Deduction in Programming

WRITE SOFTWARE WITH MATHEMATICAL PRECISION.

Written by Rohan Anil Marar

Introduction

“ Data, data, data !”, said Holmes, “I cannot make bricks without clay.”

It can be considered that Sherlock Holmes was the first data scientist, in a time when computers were not even invented. Instead, he relied on his greatest asset: his brain, which in his words, was a machine, a computer of the highest order that took input data in form of observations(meticulously analyzing crime scenes), and then performing immense calculations in his head ,rejecting impossible theorems and narrowing down to only likely possibilities, and then producing desired outcomes in his moments of ‘eureka!’.

Fast forward to 100 years from Holmes’ time, where the computer can be said to be mankind’s greatest invention, a machine more powerful than any other, simply because it can manipulate information and produce miracles.

In our modern digital age, knowing how a computer works is vital, and knowing how to control such a machine via programming is essential. Computer scientists must strive to write software with utmost perfection and mathematical beauty, and adopting Holmes’ deductive reasoning techniques while creating new algorithms may help them achieve more success.

Computer programming is all about problem-solving, and if we consider problem-solving analogous to the crime solving techniques used in detective stories, we can add a layer of fun to the tedious process of coding, as well as develop a new practical, scientific mindset.

How to develop programs with mathematical beauty:

There is a big difference between programming and coding. Even though they are assigned the same meaning many times, the difference is that whereas coding is just learning about the syntax of a given programming language like python or java, and then writing programs by employing concepts like importing functions, etc; programming includes first understanding about the science behind a software, the mathematical models employed in it,how a compiler runs that program, how data is handled and processed, how everything works inside a computer program. Coding is like learning a spoken language by only reading a dictionary, whereas programming is like learning a spoken language by conversing with people, and understanding the meaning of each and every word, and with practice, you fully grasp it.

Deductive Reasoning:

Computer Science builds its strong foundation upon mathematics. Many classical computer models are derived from complex mathematical concepts. One of the most famous model of computation, a turing machine, can be better explained by deploying lambda calculus, which also finds important applications in functional programming. Even all machine learning algorithms depend heavily on mathematical foundations for their success. Hence we can see that for understanding almost all domains related to computers, we need to have a solid grasp of the mathematics behind it.

Computer Science is broadly defined by mathematical concepts related to logic, computation, and uncertainty. Bayes theorem and other models help us with dealing with uncertainty, particularly when designing intelligent agents. Computational processes are beautifully explained by concepts like Turing machines, and many other models. Logic is best explained by many concise concepts like Type theory,etc; along with concepts which may or may not have mathematical foundations.
A particularly interesting model is deductive reasoning. Broadly speaking, it can be defined as a process of drawing inferences using logic.
Deductive reasoning starts with a simple statement and examines the possibilities to reach a specific, logical conclusion. Example: All men are mortal. Socrates is a man. Hence Socrates is mortal.

Such a train of thoughts, or the process of deduction, can find tremendous applications when designing software models or computational processes. This is because there is an inherent foundation of mathematics and logic embedded in this system, and we can employ it for any practical purposes, such as finding errors in our code, designing operating systems, finding out the most optimal strategy in a ML algorithm,etc. And we can do all this without having to delve into any complicated math behind computers, by simply changing our approach to programming, and thinking about coding like how a scientific mind such as Holmes would approach crime-solving.

Deductive profiling:

The deductive profiling method relies on the application of deductive reasoning to the observable evidence. Investigators collect general information about the crime, and the profiler draws logical conclusions about the criminal's characteristics, based on the profiler's experience,knowledge, and critical thinking. Analysis of the crime scene,forensic evidence, and behavioral analysis are all components of the deductive process.
When we bring this mindset to programming,many interesting patterns emerge.For example,consider a simple machine learning algorithm. Machine learning heavily depends on the data itself, the quality and size of the datasets,and finding

relations or connections between a certain cluster of data.The job of our algorithm is to find optimal strategies to fit in our datasets.Optimization also plays an important role in this. In case our employed algorithm goes too far away from expected results, that is,it fails to correctly optimize itself in accordance with the input data or parameters, the output generated by it will be incorrect.We call this overfitting. Hence keeping track of the algorithm’s behavior(behavioral analysis) and monitoring it regularly is vital. Often, it is up to programmers to deduce the reasons why a certain code acts in an unexpected way. Analysis of all parameters, algorithms used,etc becomes easy once you approach the problems in a deductive manner.

CHAPTER 1: WHO DID IT ? FINDING THE CULPRIT .

Detecting errors in our code, and handling them.

“Debugging is like being the detective in a crime movie where you are also the murderer”- Felipe Fortes

Errors and bugs in a computer program can be described as true crimes. They are responsible for complete failure of software, and debugging a program takes an awful lot of time. However, some errors are easy to detect and handle, while others are more notoriously hard to catch. Debugging code is not a mysterious art form. It’s like a detective solving a mystery.

Prove that a crime has been committed.

You need to determine whether there’s something that doesn’t work in your program. This is why testing code is crucial.
As you go through the debugging process, you’ll need to identify who the suspects are. Which lines of code could be

the one which committed the crime. Error handling is crucial when developing maintainable and robust code. Errors can fall into several categories: logical errors, generated errors, run-time errors, compile-time errors.

You also have witnesses in your code. Often, these are the variables containing data: what are the values of data and what type they are.

Firstly, check whether the crime has been committed. Often, when a program is compiled, errors are shown in the terminal(or console), but even then, some bugs go unnoticed.

In the words of Sherlock Holmes himself: “When you have eliminated the impossible, whatever remains,however improbable, must be the truth”. Apply this mindset when handling errors in your code. If you are stuck in your process,and cannot find the reasons why your code is not working as expected, employ the process of elimination. Eliminate reasons which you think cannot be responsible for errors. Go through your entire code again and again. Iterate this process, keep note of what you have written, so you may find something that you missed. Determine whether the error could be a compile-time error, a syntax error, etc. Keep on eliminating possible reasons, until you reach the truth.

Remember: the key is to keep on tinkering and experimenting while being patient. Programming is hard, and is a discipline

that must be followed precisely. Only after going through our program as many times as possible, we can start finding out new angles to the problem, and gain new insights.

Now, let's consider Python for our problem:

1. Print and Check

The simplest and most powerful method is to print some particular variables and check their values are as expected or not.

Code snippet:

add pseudocode here###

Here we have used the print() function as a forensic tool.

Also, using a code line in the following syntax: print(f “{variable= }”) , that is, using print() with the f-string with a “=” at the end is ideal for debugging.

If the output is not an expected answer, then that means there is some error. Hence we employ a very simple deduction trick to find out bugs in our code.

Pro tip: When using javascript, console.log() is your friend. Sometimes, errors hide under the hood, and are very hard to spot unless you print some values to the console and verify their validity.

2. Always divide your entire code in modular components

By modularizing, you can easily find errors by running each individual part of your code one by one as small, testable components. Thanks to object-oriented programming, dividing your code by creating objects,classes, functions,etc, helps a lot in such processes.

And luckily, due to the easy logical flow of object-oriented programming, where we can easily track down the control flow of a program from one object to another(as functions can communicate with each other via function calls,etc), we can construct simple logic models of our code, and easily deduce

where our code went wrong, thus detecting bugs becomes a lot easier.

Pro tip: Always employ logging when developing big software projects. Logging is just a means of keeping track of events that happens when some software is run. If you do not have a logging record and your entire program crashes, there are very few chances where you can detect the source of the problem.
Logging is like keeping track of all important facts for future uses, just like how a detective will note down everything in his log book.

3. Understanding the error encountered

When an error first appears on your screen,find the line in the error specific to your code. Lots of error messages that appear in your chosen IDE after the code is compiled are not important to the actual error itself. We have to detect the line that is the most relevant to the error. That line is generally at the top of the stack trace.(Stack trace or teaceback is just a report of the active stack frames at a curtain point in time during the program execution. A stack is just a memory allocation concept).

After you run your code, a stack trace that contains a lot of errors appears.Now, it’s up to us to deduce which line seems the most promising.

###include a code snippet here to explain above concept.###

For this step, basic knowledge of certain things is necessary. For example, you must understand what traceback means.In Python, a traceback is a report containing the function calls made in your code at a specific point,that is , when you get an error it is recommended that you should trace it backward(hence the name traceback). Whenever the code gets an exception, the traceback will give information about what went wrong in the code.
Hence we can see that this method is intuitive. Just like how a detective will follow the chain of events backwards, being vigilant to detect any new evidence along each step of the way, and reconstructing the entire history of the situation, we can follow the chain of events backwards in our program to detet bugs and find out what exactly happened.

include code snippet here to explain traceback###

4. Exception Handling in python

An exception is an event,which occurs during the execution of a program,that disrupts the normal flow of the program's instructions. In general, when a python script encounters a situation that it cannot cope with,it raises an exception.

An exception is a Python object that represents an error. Hence it is our Dr.Watson in this scenario, the one astute person that helps us in our hour of need.

When a Python script raises an exception,it must handle the exception immediately otherwise it terminates and quits.

Handling an exception:

If you have some suspicious code that may raise an exception, you can defend your program in a try: block. After using try: block,include an except: statement, followed by a code snippet that can handle the dire situation as nicely as possible.

Syntax:

```
try:
#do your operations here;

except Exception1:
If there is Exception1 here, execute this block;
.
.
.
else:
If no exception, then execute this block;
```

The above mentioned tricks work as mini-detectives for you.

5. Other important instructions to follow:

1. Never return or pass null
2. Separate error handling and main logic
3. Make a centralized exception reporter.
4.

###Elaborate or add more points in this chapter later. Write at least 3 more full pages.###

Case study: Javascript

Javascript has a built-in error object that provides error information when an error occurs.The error object provides two useful properties:name and object. “Throw, try, and catch” are the three pillars of error handling in javascript.

The *try* statement defines a code block to run (to try)

The *catch* statement defines a code block to handle any error

The *finally* statement defines a code block to run regardless of the result

The *throw* statement defines a custom error.

Hence, these four statements act as helping minions when you are trying to detect errors in your javascript code.

Chapter 2: Mind Palace: Caching,Sorting and Searching

“*The greatest undiscovered splendid wealth is hidden in the most exquisite palace in the never-ending land of your mind. You just have to find your wealth”- Debasish Midhra*

The “mind palace” is a well-known mnemonic technique that is popularized by Sherlock Holmes(in the tv show,at least). In this technique, Holmes visits an imaginary place inside his mind and performs a series of associations between words,places and objects(which further demonstrates his eidetic memory), until he arrives at the information he wants to remember. Despite appearing in a fictional story, this technique actually exists in the real world, and is known as the method of loci.

Suffice it to say that Holmes is a man who treats his brain with the utmost care, choosing only selected stuff to remember while deleting everything else from his memory. He takes immense efforts to organize all data in a completely organized, sorted manner, similar to how you would stack books on a shelf in a library. In his words: ***“I consider that a man’s brain is like a little empty attic, and you have to stock it with furniture as you choose. A fool takes in all***

the lumber of every sort he comes across,so that the knowledge which might be useful to him gets crowded out, or at best is jumbled up with a lot of other things,so that he has a difficulty in laying his hands upon it. Now the skillful workman is very careful indeed as to what he takes into his brain-attic. He will have nothing but the tools which may help him in doing his work, but of these he has a large assortment, and all in the most perfect order. It is a mistake to think that that little room has elastic walls and can distend to any extent. Depend upon it there comes a time when for every addition of knowledge you forget something that you knew before. It is of the highest importance, therefore, not to have useless facts elbowing out the useful ones."

As we can see, this is an extremely useful advice given by Holmes, and extremely relevant to computer programming. Software developers must learn how to manage memory resources in the programs they create. Memory management is the process of controlling and coordinating a computer's main memory. This is of utmost importance while designing an operating system too. But for small-time developers, having strong knowledge about the memory management of the programs they write is vital.

Memory management requires that the programmer provides ways to dynamically allocate portions of memory to programs, when requested, and free it for reuse when it is no longer

needed. In any advanced computer system,such as an operating system, where more than one process runs at any given point in time, this is critical.

In computing, a **data segment** (often denoted **.data**) is a portion of an object file or the corresponding address space of a program that contains initialized static variables, that is, global variables and static local variables. The size of this segment is determined by the size of the values in the program's source code.

The data segment is read/write, since the values of variables can be altered at run time. This is in contrast to the *read-only data segment* (*rodata segment* or *.rodata*), which contains static constants rather than variables; it also contrasts to the code segment, also known as the text segment, which is read-only on many architectures. Uninitialized data, both variables and constants, is instead in the BSS segment.

Historically, to be able to support memory address spaces larger than the native size of the internal address register would allow, early CPUs implemented a system of segmentation whereby they would store a small set of indexes to use as offsets to certain areas. The Intel 8086 family of CPUs provided four segments: the code segment, the data segment, the stack segment and the extra segment. Each segment was placed at a specific location in memory by the software being executed and all instructions that operated on the data within those segments were performed relative to the

start of that segment. This allowed a 16-bit address register, which would normally be able to access 64 KB of memory space, to access 1 MB of memory space.

This segmenting of the memory space into discrete blocks with specific tasks carried over into the programming languages of the day and the concept is still widely in use within modern programming languages.

A computer program memory can be largely categorized into two sections: read-only and read/write. This distinction grew from early systems holding their main program in read-only memory such as Mask ROM, EPROM, PROM or EEPROM. As systems became more complex and programs were loaded from other media into RAM instead of executing from ROM, the idea that some portions of the program's memory should not be modified was retained.

When any program is loaded into memory, it is organized in three parts called segments,which are:
1.Text segment (the code segment)
2.Stack segment
3.Heap Segment

Text segment

The text segment is where the compiled code of the program itself resides. This is the machine language representation of the program steps to be carried out, including all functions making up the program, both user and system defined. So, in general, an executable program generated by a compiler will have memory organization.

include figure/illustration here###

Code segment: This contains the code (executable or code binary).
The **code segment**, also known as **text segment**, contains executable code and is generally read-only and fixed size.

Data segment:The **data segment** contains initialized static variables, i.e. global variables and local static variables which have a defined value and can be modified

This is sub-divided into two parts.

- *Initialized data segment:* All the global, static and constant data are stored in the data segment.
- *Uninitialized data segment:* All the uninitialised data are stored in the Block Started by Symbol (BSS).

Stack segment

The stack is used to store our local variables, and for passing arguments to the functions along with the return address of the instruction that is to be executed after the function call is over. When a new stack frame needs to be added (as a result of a newly called function), the stack grows downwards.

Heap segment

When the program allocates memory at runtime,then memory gets allocated in heap. When some more memory needs to be allocated using certain functions, heap grows upwards.

Memory Management in Python:

Ever wondered how Python handles your data behind the scenes? How are your variables stored in memory? When do they get deleted?

Also, the most important question of all: why Python is notoriously slower than other programming languages like C++? The answer to this is simple: it is a high-level language(than C or C++) thus Python itself manages details of a program like memory allocation, memory deallocation, pointers to memory addresses, etc. Hence a part of the

reason why this occurs can be attributed to the inherent processes involved during memory allocation.

Understanding memory allocation is important to any software developer as writing efficient code means writing a memory-efficient code. Memory allocation can be defined as allocating a block of space in the computer memory to a program. In Python memory allocation and deallocation method is automatic as the Python developers created a garbage collector for Python so that the user does not have to do manual garbage collection.
Garbage collection is a process in which the interpreter frees up the memory when not in use to make it available for other objects.
Assume a case where no reference is pointing to an object in memory i.e. it is not in use so, the virtual machine has a garbage collector that automatically deletes that object from the heap memory

Reference counting works by counting the number of times an object is referenced by other objects in the system. When references to an object are removed, the reference count for an object is decremented. When the reference count becomes zero, the object is deallocated.
For example, Let’s suppose there are two or more variables that have the same value, so, what Python virtual machine does is, rather than creating another object of the same value

in the private heap, it actually makes the second variable point to that originally existing value in the private heap. Therefore, in the case of classes, having a number of references may occupy a large amount of space in the memory, in such a case referencing counting is highly beneficial to preserve the memory to be available for other objects

Example: Code snippet in Python:
include code example here

Python uses a portion of the memory for internal use and non-object memory. Another part of the memory is used for Python object such as int, dict,list,etc. Memory management in Python involves a private heap containing all Python objects and data structures.
The management of this private heap is ensured internally by the Python memory manager.

Memory Allocation in Python

There are two parts of memory:

- stack memory
- heap memory

The methods/method calls and the references are stored in stack memory and all the values objects are stored in a private heap.

Work of Stack Memory

The allocation happens on contiguous blocks of memory. We call it stack memory allocation because the allocation happens in the function call stack. The size of memory to be allocated is known to the compiler and whenever a function is called, its variables get memory allocated on the stack.
It is the memory that is only needed inside a particular function or method call. When a function is called, it is added onto the program's call stack. Any local memory assignments such as variable initializations inside the particular functions are stored temporarily on the function call stack, where it is deleted once the function returns, and the call stack moves on to the next task. This allocation onto a contiguous block of memory is handled by the compiler using predefined routines, and developers do not need to worry about it.

Example:

```
 def func():
#all these variables get memory allocated on stack
a=20
b=[]
c= " "
```

Work of Heap memory:

This memory is allocated during execution of instructions written by programmers.Note that the name heap has nothing

to do with the heap data structure. It ia called heap because it is a pile of memory space available to programmers to be allocated or de-allocated. The variables are needed outside of method or function calls or are shared within multiple functions globally are stored in Heap memory.
Example:
#this memory for 10 integers is allocated on heap
a=[0]*10

Memory is an empty book:

You can begin by thinking of a computer's memory as an empty book intended for short stories. There's nothing written on the pages yet. Eventually, different authors will come along. Each author wants some space to write their story in.

Since they aren't allowed to write over each other, they must be careful about which pages they write in. Before they begin writing, they consult the manager of the book. The manager then decides where in the book they're allowed to write.

Since this book is around for a long time, many of the stories in it are no longer relevant. When no one reads or references the stories, they are removed to make room for new stories.

In essence, computer memory is like that empty book. In fact, it's common to call fixed-length contiguous blocks of memory pages, so this analogy holds pretty well.

The authors are like different applications or processes that need to store data in memory. The manager, who decides where the authors can write in the book, plays the role of a memory manager of sorts. The person who removed the old stories to make room for new ones is a garbage collector.

Memory Management: From Hardware to Software

Memory management is the process by which applications read and write data. A memory manager determines where to put an application's data. Since there's a finite chunk of memory, like the pages in our book analogy, the manager has to find some free space and provide it to the application. This process of providing memory is generally called memory allocation.

On the flip side, when data is no longer needed, it can be deleted, or freed. But freed to where? Where did this "memory" come from?

Somewhere in your computer, there's a physical device storing data when you're running your Python programs. There are many layers of abstraction that the Python code goes through before the objects actually get to the hardware though.

One of the main layers above the hardware (such as RAM or a hard drive) is the operating system (OS). It carries out (or denies) requests to read and write memory.

Above the OS, there are applications, one of which is the default Python implementation (included in your OS or

downloaded from python.org.) Memory management for your Python code is handled by the Python application. The algorithms and structures that the Python application uses for memory management is the focus of this article.

The Default Python Implementation

The default Python implementation, CPython, is actually written in the C programming language.

The Python language is defined in a reference manual written in English. However, that manual isn't all that useful by itself. You still need something to interpret written code based on the rules in the manual.

You also need something to actually execute interpreted code on a computer. The default Python implementation fulfills both of those requirements. It converts your Python code into instructions that it then runs on a virtual machine.

Note: Virtual machines are like physical computers, but they are implemented in software. They typically process basic instructions similar to Assembly instructions.

Python is an interpreted programming language. Your Python code actually gets compiled down to more computer-readable instructions called bytecode. These instructions get interpreted by a virtual machine when you run your code.

Have you ever seen a .pyc file or a __pycache__ folder? That's the bytecode that gets interpreted by the virtual machine.

Okay, so CPython is written in C, and it interprets Python bytecode. What does this have to do with memory management? Well, the memory management algorithms and structures exist in the CPython code, in C. To understand the memory management of Python, you have to get a basic understanding of CPython itself.

CPython is written in C, which does not natively support object-oriented programming. Because of that, there are quite a bit of interesting designs in the CPython code.

You may have heard that everything in Python is an object, even types such as int and str. Well, it's true on an implementation level in CPython. There is a struct called a PyObject, which every other object in CPython uses.

Memory management is an integral part of working with computers. Python handles nearly all of it behind the scenes, for better or for worse.

Python abstracts away a lot of the gritty details of working with computers. This gives you the power to work on a higher level to develop your code

without the headache of worrying about how and where all those bytes are getting stored.

Caching:

caching: a mechanism that minimizes unnecessary computations and speeds up your programs. You'll be surprised by how effective this is when done right.

What is caching?

Caching is an optimization technique that consists in keeping recently (or frequently) used data in a memory location that has cheap and fast access for repeated queries.

Due to multiple reasons, accessing the data from its original source can be expensive and caching appears as a solution that alleviates this problem.

👉 Let's consider an application where caching is commonly used: web servers.

You're building a small website to share local news in your area.

When a user scrolls the newsfeed and clicks on a post, he's redirected to a web page: what the browser does at this step is querying a remote server, receiving the page source code and rendering it in a human-readable format. As you may expect, this operation is time-consuming because it involves downloading remote files (it's a network-bound operation) and rendering them.

⚠️ Repeating this same operation each time the user clicks on this link seems like an unnecessary computation. We already know the result after the first fetch, why not reuse it?

✅ What you should do in this situation is storing the content locally after fetching each article. The next time the

user opens the same article, the app will read the content from the local copy. This will be way faster.

You've probably implemented caching without even knowing it

Let's get back to the previous example and try to come up with a naive implementation of caching.

What we want is to store the content of each post in local memory (an object in RAM for example) and reusing it later if the user requests the same link later.

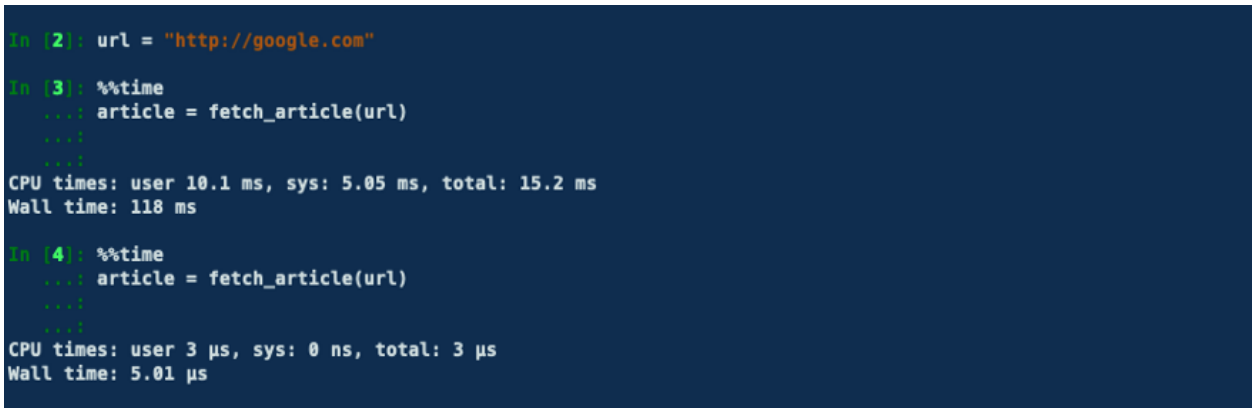
This looks like a perfect job for dictionaries.

```
import requests


# initialize cache a the beginning
```

```
cache = dict()

def extract_article_content(url):

    response = requests.get(url)

    content = response.content

    return content



  def fetch_article(url):

    if url not in cache:

        content = extract_article_content(url)
```

```
    cache[url] = content

return cache[url]ort requests
```

If we try to run `fetch_article` two consecutive times with the same url.



Screenshot by the author

We'll notice a huge difference.

- **First time: 118 ms**

- **Second time: 5.01 µs**

That's a whopping ratio of 2360!

118 ms is still fast, you'd tell me. But imagine if the same user does this operation multiple times per day. There's a lot of time to save.

Sorting Algorithms in Python

Sorting is defined as an arrangement of data in a certain order. Sorting techniques are used to arrange data(mostly numerical) in an ascending or descending order. It is a method used for the representation of data in a more comprehensible format. It is an important area of Computer Science. Sorting a large amount of data can take a substantial amount of computing resources if the methods we use to sort the data are inefficient. The efficiency of the algorithm is proportional to the number of items it is traversing. For a small amount of data, a complex sorting method may be more trouble than it is worth. On the other hand, for larger amounts of data, we want to increase the efficiency and speed as far as possible. We will

now discuss the several sorting techniques and compare them with respect to their time complexity.

Sorting Algorithms:

###write more about bubble sort, etc….###

Searching in Python:

Searching is a very basic necessity when you store data in different data structures. The simplest approach is to go across every element in the data structure and match it with the value you are searching for.This is known as Linear search. It is inefficient and rarely used, but creating a program for it gives an idea about how we can implement some advanced search algorithms.

Linear Search

In this type of search, a sequential search is made over all items one by one. Every item is checked and if a match is found then that particular item is returned, otherwise the search continues till the end of the data structure.

Example

```
def linear_search(values, search_for):
   search_at = 0
   search_res = False
# Match the value with each data element
   while search_at < len(values) and search_res is False:
      if values[search_at] == search_for:
         search_res = True
      else:
         search_at = search_at + 1
   return search_res
l = [64, 34, 25, 12, 22, 11, 90]
print(linear_search(l, 12))
print(linear_search(l, 91))
```

Output

When the above code is executed, it produces the following result –

```
True
False
```

Interpolation Search

This search algorithm works on the probing position of the required value. For this algorithm to work properly, the data collection should be in a sorted form and equally distributed.Initially, the probe position is the position of the middle most item of the collection.If a match occurs, then the index of the item is returned.If the middle item is greater than the item, then the probe position is again calculated in the sub-array to the right of the middle item. Otherwise, the item is searched in the subarray to the left of the middle item. This process continues on the sub-array as well until the size of subarray reduces to zero.

Example

There is a specific formula to calculate the middle position which is indicated in the program below –

```
def intpolsearch(values,x ):
  idx0 = 0
  idxn = (len(values) - 1)
  while idx0 <= idxn and x >= values[idx0] and x <= values[idxn]:
# Find the mid point
        mid = idx0 +\
    int(((float(idxn - idx0)/( values[idxn] - values[idx0]))
    * ( x - values[idx0])))
# Compare the value at mid point with search value
  if values[mid] == x:
    return "Found "+str(x)+" at index "+str(mid)
  if values[mid] < x:
    idx0 = mid + 1
  return "Searched element not in the list"

l = [2, 6, 11, 19, 27, 31, 45, 121]
print(intpolsearch(l, 2))
```

Output

When the above code is executed, it produces the following result –

```
Found 2 at index 0
```

Searching Algorithms

Searching Algorithms are designed to check for an element or retrieve an element from any data structure where it is stored. Based on the type of search operation, these algorithms are generally classified into two categories:

1. **Sequential Search: In this, the list or array is traversed sequentially and every element is checked. For example: Linear Search.**
2. **Interval Search: These algorithms are specifically designed for searching in sorted data-structures. These type of searching algorithms are much more efficient than Linear Search as they repeatedly target the center of the search structure and divide the search space in half. For Example: Binary Search.**

include many more illustrations, code , and descriptions in this chapter. I have just copy-pasted many stuff from other websites in this chapter, so i will first have to edit and write original stuff for many pages in this chapter, to avoid copywrite accusations. Also simplify many stuff in this chapter, since some concepts are a little tough to understand###

Chapter 3: Hypotheses : Using Machine Learning for awesome predictions.

“ It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories instead of theories to suit facts.”- Sherlock Holmes, A study in Scarlet

There is a reason why data is called the new oil. In the 21st century, we are currently in the Information Revolution. The economy of the entire world is fueled by communication systems, information gathering, and data collection. Some governments rely heavily on information and even consider it as a potential weapon, and one of the most useful assets. For it to possess tremendous value, raw data must be broken down and analyzed to glean useful information from it, using data analytics.

Machine Learning is the epitome of Data Science. Machine learning essentially uses data to adjust algorithms and make them better,while at the same time is used to make useful predictions. Some computer scientists go so far as to say that data is much more important than the algorithms used, when it comes to machine learning. Data is given a higher priority, and quality and amount of data used for computations is extremely vital.

Enter Sherlock Holmes, the detective who uses his deductive reasoning powers to crunch abstract random pieces of data(facts and evidence), and uses it to detect hidden patterns to solve the mystery. This is remarkably similar to how a Machine learning algorithm works: it takes input data fed to it, once the input datasets are cleaned and analyzed properly, then it adjusts many parameters and variables according to the data points in order to find hidden valuable patterns between the different data points, all the while it tries to optimize the algorithm to better suit the data itself. There is a strong correlation between accuracy of deduction and inferences, and the quality of data available.

The performance of a ML algorithm greatly depends on the input data provided to it. If you feed it inaccurate or worthless data, it will provide worthless output. Not all data is created equal. Real world data has high chances of being distorted with unwanted noise,etc; hence data analytics is extremely important before feeding data to an algorithm. The process of making raw data available in consumable state is called ‘feature engineering’.

One should never approach a problem with preconceived notions or bias. As Holmes says: ***“Let me run over the principal steps. We approached the case, you remember, with an absolutely blank mind, which is always an advantage. We had formed no theories. We were simply there to observe and to draw inferences from our observations.”*** Similarly, in

the realm of Machine learning, we should never include bias,prejudices or preferences in the data that we use, or else the output of the ML algorithm will also turn out to be heavily biased. Unlike some expert systems used in AI, Machine learning is all about learning independently without need of any explicit programming. A typical ML algorithm carefully analyzes the data being fed, identifies useful patterns, makes predictions or inferences ,while at the same time adjusting its parameters to iteratively improve its model performance by learning everytime new data is presented to it. All the steps mentioned above are exactly what a forensic specialist like Sherlock Holmes would do, while solving crimes.

We can thus safely say that Machine Learning is the Sherlock Holmes of Data science!

Machine Learning : It is a branch of Artificial Intelligence which focuses on the use of data and algorithms to imitate the way the human brain works and learns, gradually improving its accuracy.
Note that it a subfield of AI. AI by itself is a broader field, with many other concepts like reinforcement learning, game theory, probability,etc; included in it.
Some of the best models involved in Machine learning comprise of modeling neural networks(imitating human brain). Neural networks are implemented in software(algorithm) form by use of very sophisticated mathematical models. In short, the foundation of Machine learning heavily relies on mathematical models and concepts. For example, logistic regression, a common ML algorithm, employs the sigmoid function commonly used in math, to train the neural network and adjust the parameter values,etc.
However, software developers need not fear. In their coding life, they can easily get used to playing with ML without having to go through all the complex math stuff or theoretical concepts. This is the power of computer science: abstraction. We can stand on the shoulders of giants(who designed the complex algorithms) to create new stuff without having to bother with the complicated technicalities.

Feature Engineering in ML:

Raw data presents a serious challenge. Raw data is often unfiltered, contains outliers or misleading values,and a lot of noise. If we feed garbage data to an algorithm, we will get garbage data as our output. This is why feature engineering is used: to create features, that is, attributes or properties in our dataset, that make a ML algorithm work. For example, in spam detection algorithm used in our email system, features may include aspects of the data such as presence or absence of a certain word, which is decided and labeled by the algorithm as spam.
Notice how this is extremely similar to Holmes deducing about Watson's lifestyle changes by observing data elements(that is, changes in his appearance).

Feature engineering is the first step that programmers use when they are designing a ML algorithm. It is a machine learning technique that leverages data to create new variables that are not in the input dataset. Input dataset comprises features, which are in the form of structured columns.

Feature engineering has 2 main goals:

1. Preparing the proper input dataset, compatible with the Ml algorithms.
2. Improving performance of machine learning models.

According to a survey in Forbes, data scientists spend 80% of their time on data preparation.

One of the best advantages of ML is that we can graphically visualize the shape and size of the datasets. This step simplifies everything for us.Also, we can assign different values of data as each individual data ‘point’. Simply imagine points on a graph. These are the values in your datasets.

Steps in Feature engineering:

1. **Handling Outliers:**

   An outlier is a data point that is noticeably different from the rest. Hence when we use a ML algorithm on that dataset, this value will cause errors in the output. Hence we must deal with such outliers first.
   If a value has a distance to the average higher than x*standard deviation, then that value is an outlier.

   Step 1: Importing necessary packages

```
import numpy as np
Import pandas as pd
Import matplotlib.pyplot as plt
Import seaborn as sns
```

   Step 2: load the dataset

```
df=pd.read_csv(‘random.csv’)
```

df.sample(5);

CHAPTER 5: INFILTRATING THE CRIME SYNDICATE: UNDERSTANDING SYSTEM DESIGN

“*It is impossible to design a system so perfect that no one needs to be good”-* T.S. Elliot

What is a system: It is an orderly grouping of interdependent components linked together according to a plan to achieve a specific goal.
A system must have some structure and behavior.
Interconnectivity and interdependence must exist among the

system components. And the objectives of the organization must have higher priority than the objectives of the subsystems.

System design is an extremely important concept in computer programming. Almost all software developers encounter this domain at some point in their careers, simply because it is essentially the backbone of all existing software products, and how they are integrated in the broader realm of client-server systems,etc. Hence understanding this concept is very essential, and this process could be made more fun by tackling it using the mindset of a detective, and imagining an immense computer system as a big mafia organization having individual working components.

In honor of the greatest rival of Sherlock Holmes, we must of course bring **Professor Moriarty** for our analogies in the context of programming. Hence in the context of system design, the control element of the system which is the decision-making subsystem that controls the pattern of activities governing input, processing, and output,can be effectively called as our Professor Moriarty. He is both the control, and the processor, in our model of system design.

Holmes describes Moriarty as: *“ a spider in the center of its web, but that web has a thousand radiations, and he knows well how to quiver each of them. He does little himself. He only plans. But his agents are numerous and splendidly*

organized." Arthur Conan Doyle, the author, used the term "spider's web" to describe the complex network of Moriarty's criminal empire with Moriarty himself being the one in control.

Let's see how we can use analogies to understand system design:
Firstly, a system must have the following properties:

1.**Organization**: This implies structure and order. It is the arrangement of components that helps to achieve predetermined objectives. As mentioned earlier, the objectives of the entire system(for example: a network of interconnected computer servers) are given higher priority than the objectives of individual components(for example: subsystems executing their individual processes).
Here, in the grand scheme of objectives, the crime syndicate(main system or architecture) led by Professor Moriarty hires many small-time street criminals or gangs(which are the individual components). There is a defined chain of command (with Moriarty at the top), and all major objectives of this organization are delegated, or passed down, among the respective components which then carry out their allotted tasks to further the goals of the main system.

2.**Interaction and interdependence**: Components of a system depend on each other for their normal operation.The output of one subsystem is required by other subsystem as input. Similarly, whenever an important piece of information or

fact is found out by one of the agents in the syndicate, it is then communicated upwards through the chain of command via the immense network.

In short, the objective of a system must be central. It may be real or stated. It is not uncommon for an organization to state an objective and operate to achieve another.

However, some aspects of cloud computing take this a step further, by introducing the concept of distributed and decentralized systems. Now, instead of having one Moriarty as a single computer server handling all the operations, we now have a collection of mini-Moriarties, each having individual control over their respective computational processes, in a wide network of decentralized systems.Or more simply, there need not even be a Moriarty for the system to operate. This implies that even if one computer crashes or stops working,the network system itself will not get affected, and work will continue. Hence even without one Moriarty, the system is so well established that computations continue, and no data or time is lost. We also know that even after Moriarty was taken out by Holmes, his empire was affected a lot, but was not entirely broken down. The workings of the established system was continued by the many agents(minions of Moriarty), even after their leader was eliminated. We can bring this analogy to understand microservices in system design, a point I will explain later.

However, even if all aspects of system design can be explained in a model of an organization or architecture, the flow of data and computational processes between individual components in the wide network of the system becomes hard to track down and understand. We can then rely on our old friend to help us: deduction.For example, if you cannot understand how APIs work when communicating between two services, just think logically: what does one service gain from another when the two of them communicate? What could be the main benefits of introducing an interface between two processes? Brainstorming like this while solving such problems with the mindset of a detective helps us to simplify everything. Understand the relations between the processes involved, by first obtaining facts, then forming hypotheses. When you lack source of information,simply rely on deduction.

Getting started with system design:

First, we will consider a common question on system design asked during technical interviews. We will consider: **URL shortening service**. This is an important example of system design.
URL shortening services like bit.ly or TinyURL are very popular to generate shorter forms for long URLs. You need to design a system , a web service, where if a user gives a long URL then the service returns a short URL, and if the user gives a short URL then the service returns the original URL.

For example:

include pseudocode here

The objective of the system is that when a user gives a long URL, it converts it into a short version and updates the database involved, and the reverse process when user enters short URL.This kind of problem is difficult because we have to consider the scalability and durability of the service.

Requirements of the system:

Difference between centralized, decentralized , and distributed systems:

Other concepts in system design explained using deductive reasoning:

1. **Load balancing:**
2. **Microservice vs monolith**
3. **APIs**
4. **Horizontal and vertical scaling**
5. **Proxies**
6. **Caching**
7. **CDN**
8. **Message queue**
9. **Network protocols**
10. **Containerization**
11. **Indexing and sharding**