**Applied Machine Learning Notebook** 

Source: <https://www.appliedaicourse.com/course/11/Applied-Machine-learning-course>

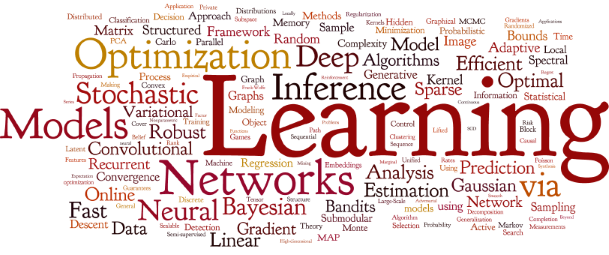
Applied Machine Learning Course

**The course that gets you HIRED!**

Notebook Written by: Rohit Kumar Jha

Enrolled Date: 4th of July, 2021

Completed On : Yet to be completed



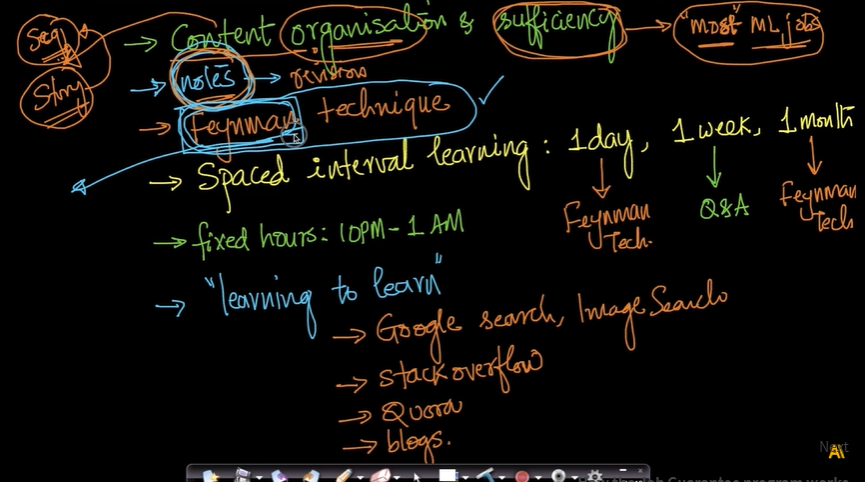
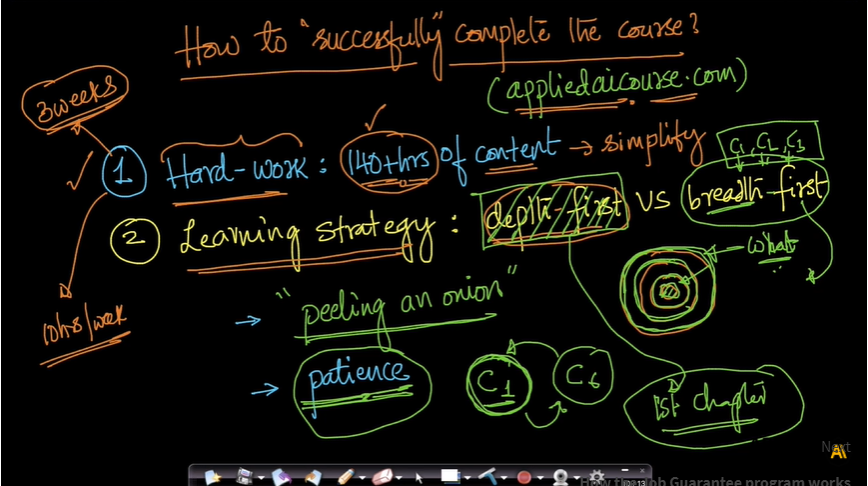
**Module 1: Fundamentals of Programming**

**How to utilise Applied AI Course:**

This course is of 200 hrs long, many people completed this course in 3-months’ time frame. So, It is very much possible that one can finish this course is 3 months.

For any long and important work need planning and proper execution. The time which you have must be utilise in effective manner. So, as Chetan Bhagat said in one interview that, you do many work in your day-to-day life which might be difficult also but those are small work, and anyone can do that work. Hence, no one respect that work (Choti Mahnat ke kaam). On the other hand, some works are big which needs consistency for longer run, which requires months of hard work that is called as Badii Mahnat, and this comes with huge respect as it is something which only few people can do because it requires effort for long duration.

So, if you want to eat an Elephant, how you eat that elephant it is very much sure that it can’t be eaten in one go, you will eat it in small-small pieces for many days and ultimately you ate a compete Elephant, and eating small piece is very easy and it doesn’t require lots of effort. Same goes with important work.



**Chapter 2: Python for Data Science Introduction**

2.1 **Python, Anaconda and relevant packages**

Link: <https://www.youtube.com/watch?v=hbUJ6nd-9lA>

<https://repo.continuum.io/archive/>

2.2 **Why learn Python?**

Python is very simple to pick up; It has sudo language kind of syntax and lots of libraries.

Packages useful for ML are available in Python.

Jupyter Notebooks for interactive programming.

Extensively used in the industry.

Python is much more general-purpose programming language.

2.3 **Keywords and identifiers**

**Keywords** are the reserved words in python.

We cannot use a keyword as variable name, function name or any other identifier.

Keywords are case sensitive.

Import keyword

Print(keyword.kwlist)

Example: False, None, True, class, if, else, return, def, try, while, for, etc

Print(“Total number of keywords : “,len(keyword.kwlist));

**Total number of keywords: 33**

**Identifiers:**

Name given to entities like class, functions and variables

Can be a combination of letters, digits and underscores, cannot start with a digit

Keywords cannot be used as identifiers, special characters cannot be used

Python has straight forward Error indications.

2.4 **Comments, indentations and statements**

Start a line with a # or use triple quotes, ‘’’ ‘’’

Indentations are used (4 spaces preferred) to make blocks of code, a for loop

Rather than writing code in a single line try to write in multiple lines (can use \) to make code readable

The written instructions are called statements

2.5 **Variables and data types in python**

**Variable** is a location in memory used to store some data; Variable declaration is not needed

a, b = 10, ‘Hi’

id(a) prints location of a

**Data types:**

Everything in python is an **object**.

**Number**: Integers, float and complex

**Boolean:** True and False

**Strings:** Sequence of Unicode characters, defined with quotes, indexable, sliceable

**List:** An ordered sequence of items, like an array, can have multiple data type elements, defined with square brackets; Lists are mutable

**Tuple:** Defined with parenthesis, can have multiple data type elements, tuple is immutable (can’t be changed once created), can be indexable

**Set:** Defined with Curly braces, Set is an unordered collection of unique items; behaves as a set-in mathematics; does not support indexing

**Dictionary:** an unordered collection of key-value pairs, defined with curly braces and a colon, value accessible with key

Data types can be converted provided the value is valid in both data types.

List(‘Hello’) = [‘H’,’e’,’l’,’l’,’o’]

2.6 **Standard input and output**

Output: print()

print(‘ {} {}’.format(a,b))

print(‘ {a} {b}’.format(a = 1, b = 2))

Input: input()

2.7 **Operators**

Operators are special symbols in python that allow arithmetic or logic computation.

2 + 3 : + is an operator and 2, 3 are operands

**Types:** Arithmetic, Comparison, Logical, Bitwise, Assignment, Special

**Arithmetic:** +, -, \*, /, %, //, \*\* (addition, subtraction, multiplication, division, modulo division, floor division, exponent)

15, 2: +: 17, - : 13, \*: 30, /: 7.5, %: 1 (remainder), //: 7, \*\*: 225;

-15//2 = -8

**Comparison:** <, >, !=, ==, >=, <=

**Logical:** and, or, not

**Bitwise:**

a= 10, b = 4:

a & b: 1010 & 0100: 0000 (and) = 0

a | b: 1110 = 14

or: |, not: -, xor: ~, rightshift: >>, leftshift: <<

a>>b:

**Assignment operator**:

=, +=, -=, \*=, /=, %=, //=, \*\*=, &=, |=

a += 10: a = a + 10

**Identity operators:**

is, is not

**Membership operators:**

In, not in

2.8 **Control flow:** if else

If test expression:

statement(s)

Example: num = 10

if num>0:

print(“number is positive”)

elif num ==0:

print(“zero”)

else:

print(“number is negative”)

print(“always printed”)

2.9 **Control flow:** while loop

**While loop:** block of code runs until a test expression is true;

lst = [10, 20, 30, 40, 50]

index = 0

while index < len(lst):

product \* = lst[index]

index += 1 # increment statement is important

0: 1\*10, 1: 10\*20, 2: 200\*30,…

We can use an else block when the test condition fails;

2.10 **Control flow**: for loop

Used to iterate over a sequence;

for element in sequence:

statement(s)

for ele in lst:

product \*=ele

**range():** range(10) will generate a list of 10 numbers from 0 to 9;

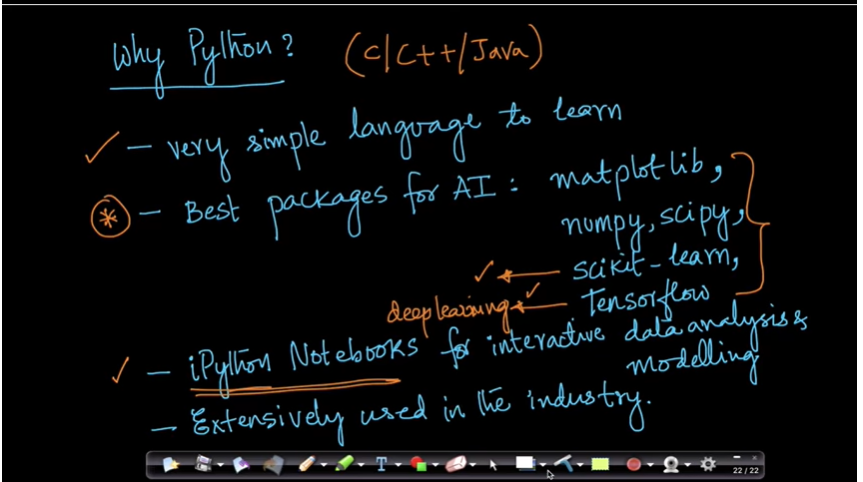
2.11 **Control flow**: break and continue

for var in sequence:

if condition:

break

Why learn Python?



**Chapter 3: Python for Data Science: Data Structures**

3.1 **Lists**

Data Structures: collection of data elements

List: Sequence/ordered data structures, these are indexable, mutable, defined by square brackets and elements are comma separated

**Operations on list:**

* + - len(list), append(element), insert(index, element), remove(element) (removes first occurrence only), list.append(element), list.extend(list), pop(index)
    - [‘one’, ’two’].append([‘one’, ’two’]) = [‘one’, ’two’, [‘one’, ’two’]]
    - [‘one’, ’two’].extend([‘one’, ’two’]) = [‘one’, ’two’, ‘one’, ’two’]
    - del lst[1]
    - list reverse: list.reverse()
    - sorted(list), list.sort()
    - lst = [1, 2, 3, 4, 5]; abc = lst; abc.append(6); print(lst) 🡪 [1, 2, 3, 4, 5, 6]
      * lst and abc are pointers;
    - string.split(‘ ‘)
    - lst[index]
    - lst[slice\_index\_start: slice\_index\_end]
    - lst1 + lst2
    - lst.count()
    - for ele in lst: print(ele)
    - **List comprehensions:** [i\*\*2 for i in range(10) if i%2 ==0]
      * [[row[i] for row in matrix] for i in range(4)]

3.2 **Tuples part 1**

A tuple is similar to list; Tuple is immutable, its elements cannot be altered.

* T = “abcd”, # comma is important to create a tuple
* Tuple access: T[1]
* Changing a tuple: a list in a tuple is mutable;
* Concat tuples using +

3.3 **Tuples part 2**

- Deletion: whole tuple will be deleted

- Tuple.count(), tuple.index(element), element in tuple, element not in tuple, len(tuple), sorted(tuple), min(tuple), max(tuple), sum(tuple)

3.4 **Sets**

Sets are unordered collection of unique items; Mutable, non-indexable.

S = {1, 2, 3}

Sets does not allow duplicate numbers.

set([1,2,3,1]) = (1, 2, 3)

Operations: set.update(elements or sets), set.discard(element), set.remove(element), set.pop(), s.clear()

Set1 | Set2: Union; Set1.union(Set2)

Set1 & Set2, Set1.intersection(Set2)

Set1 – Set2, Set1.difference(Set2)

Set1^Set2, Set1.symmetric\_difference(Set2)): Union - Intersection

Frozenset: immutable sets: Set1 = frozenset([1,2,3,4])

3.5 **Dictionary**

An unordered collection of key value pairs;

O(1) for time complexity for search tasks;

Dictionary is mutable;

Operations: dict.pop(key), dict.clear(), dict.fromkeys(list, values), dict.items(), .keys(), .values(), .copy()

Dictionary Comprehension:

for pair in d.items(): print(pair)

{k:v for k,v in d.items() if v>2}

{k:v for k+’c’,v\*2 in d.items() if v>2}

3.6 **Strings**

Sequence of characters: (Unicode (default) or ASCII)

S = “kl” or = str(1)

Access characters of a string as a list;

Strings are immutable;

Operations: str1+str2; for i in string: \_\_\_\_\_\_;

lower(), upper(), join(), split(), find(), replace()

“Bad Morning”.replace(“Bad”, “Good”)

Palindrome:

Mystr = “MaDam”

Mystr = Mystr.lower()

revStr = reversed(Mystr)

if list(Mystr) == list(revStr):

print(“Palindrome”)

else:

print(“Not palindrome”)

Alphabetic sort:

Word = word.split().sort()

**Chapter 4: Python for Data Science: Functions**

4.1 **Introduction**

Functions: a group of related statements that perform a specific task;

Converts a program into smaller chunk which makes management easy

def function():

‘’’

Doc string

‘’’

statements

return

Doc strings is written to explain the working of the function (function.\_\_doc\_\_)

Scope and Life Time of Variables: Portion of the code where the variable is recognized and Lifetime is the period throughout which the variable exists in memory

Variable inside a function are local variables which are destroyed once the function finishes execution; Global variables are not destroyed unless deleted;

Program to print highest common factor (HCF):

def computeHCF(a, b):

“””

Computing HCF of two numbers

“””

Smaller =b if a>b else a

hcf = 1

for i in range(1, smaller + 1):

if (a%i==0) and (b%i==0):

hcf = i

return hcf

4.2 **Types of functions**

**Built-in Functions and User defined functions.**

Built-in: abs(), all(), any(), dir(), divmod(), enumerate(), filter(), map(), reduce(), isinstance(),

enumerate(): returns a list with an index

filter(): applies a function on a list to reduce the list

map(): applies a function on all items of a list

def PowerOfTwo(num):

Return num\*\*2

map(PowerOfTwo, list)

reduce(): applies a computation and returns a result;

4.3 **Function arguments**

Functions need inputs which take in values through arguments;

Default arguments: to give default values to a function; have default arguments at the end

Keyword arguments: variable number of arguments can be given as input;

Arbitrary arguments: Used when number of arguments are unknown given as input to the function;

4.4 **Recursive functions**

Function calling inside itself;

Factorial(n) = n\*Factorial(n-1)

Stack of function calls; Makes code clean, hard to debug;

4.5 **Lambda functions**

Functions without name; used along with filter and map

Def Double(x):

return x\*2

Can also be defined as:

Double = lambda x : x\*2

Example: list(filter(lambda x : (x%2==0), [1,2,3,4,5])); Output: [2, 4]

reduce(lambda x, y : x\* y, [1, 2, 3, 4, 5])

4.6 **Modules**

Module refers to file containing statements and definitions.

A .py file containing code that is used in other programs

Module: example.py

import example

* import math
* math.pi

import math as m

import datetime

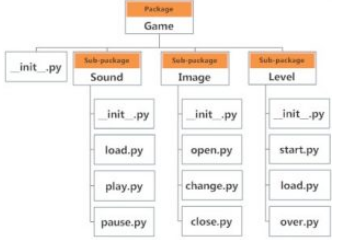
Module is basically a file which contains classes and functions.  
Package is a kind of directory which contains modules of similar type.  
Library is a collection of Packages.  
Framework is collection of Libraries.

Use dir() to get all names functions inside the module

4.7 **Packages**

\_\_init\_\_.py needs to be present in a folder to consider the folder as a package.

Packages contain modules.



import Game.Sound.play

4.8 **File Handling**

Storing data in hard disk which is non-volatile:

Open, read and close

File = open(‘example.txt’)

File can be open in different modes:

‘r’, reading (default)

‘w’, writing (creates a new file, if exists deletes it)

‘a’, appending

Closing a file:

File.close() # to make sure the data is not volatile

Use safer “try finally” block: Exception handling

import os

os.mkdir(‘test’)

os.rmdir(‘test’) # does not remove if test is not empty

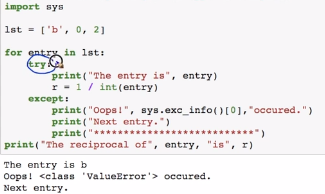
import shutil

shutil.rmtree(‘test’) # removes non-empty folders

4.9 **Exception Handling**

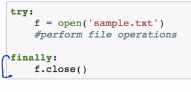
Whenever an error is observed, python interpreter raises an error;

Use:



Raising exceptions:

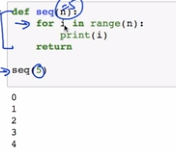
For example: Memory error

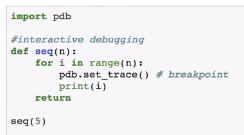


Finally block runs at the end of all operations, such as closing the file even if the file is not written, to save data.

4.10 Debugging Python

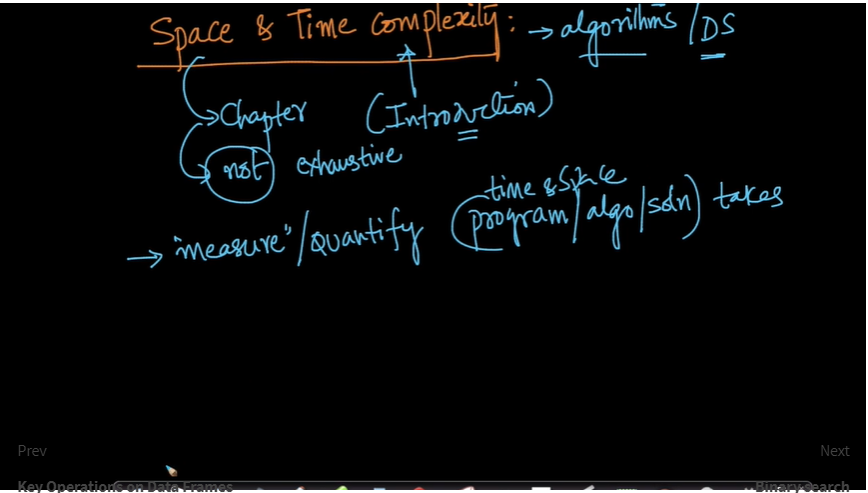
Python debugger: pdb

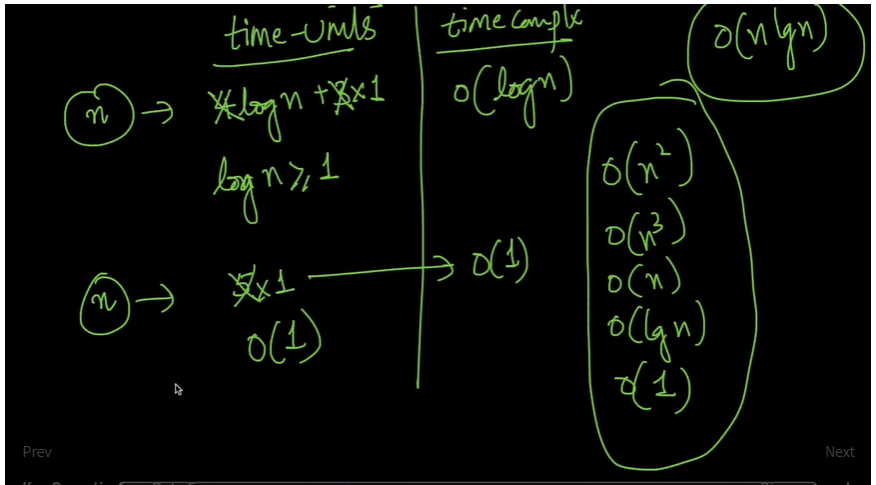
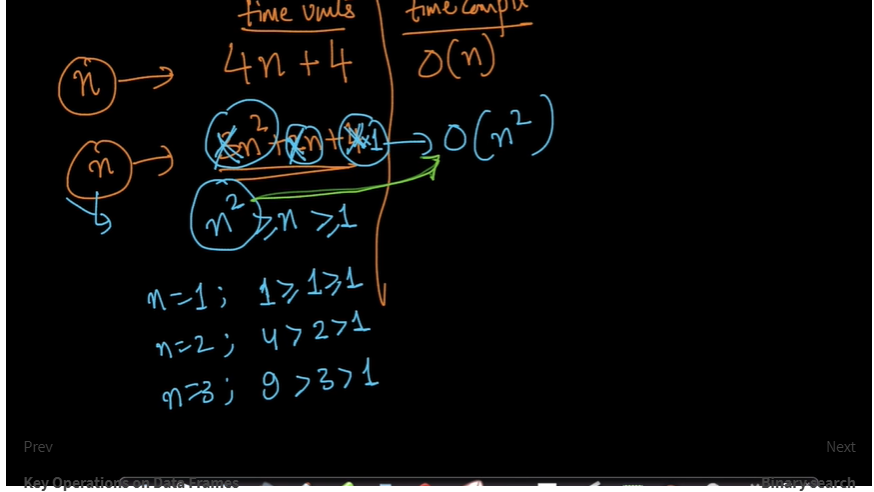
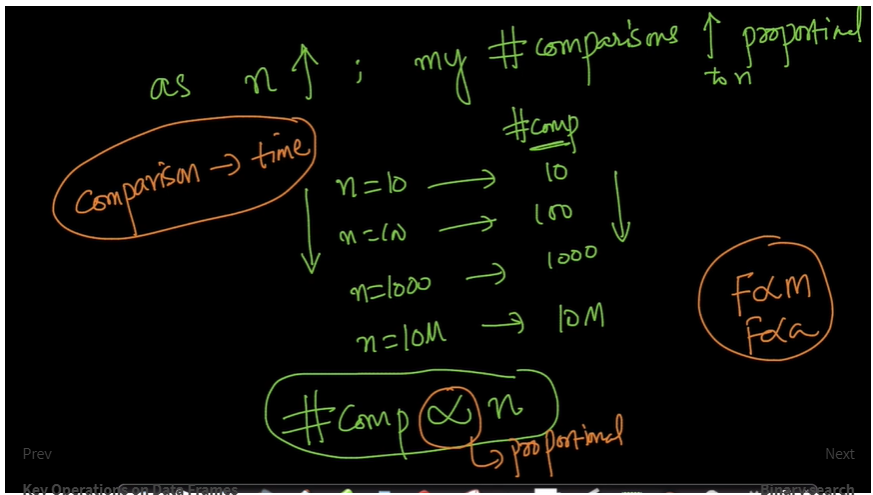
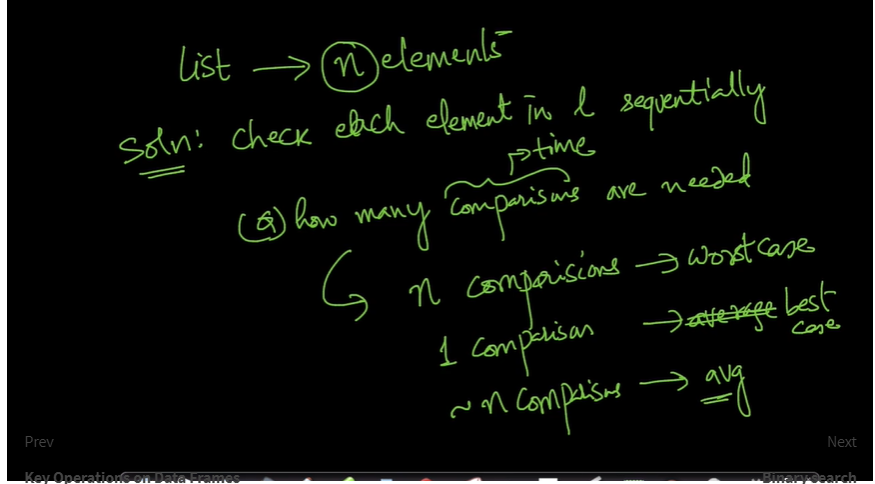
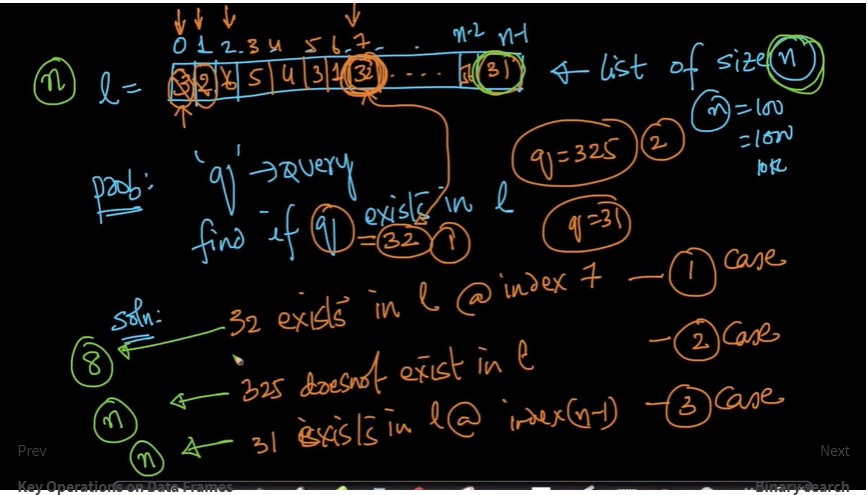




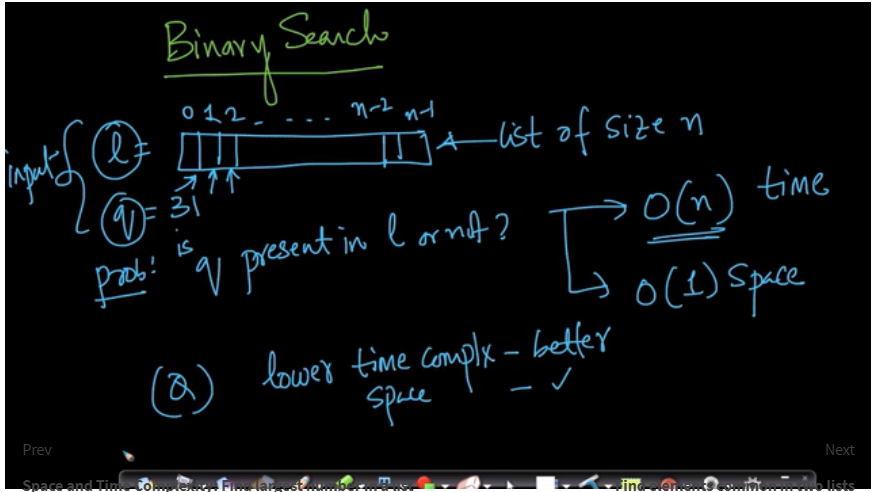
Comes to pdb.set.trace at every iteration, requires string input to give results

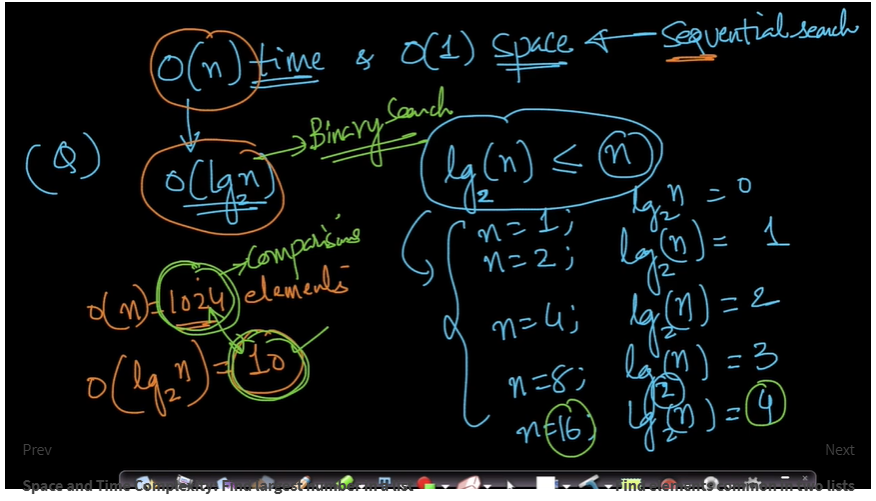
[Space and Time Complexity: Find largest number in a list](https://beta.appliedaicourse.com/lecture/11/applied-machine-learning-online-course/2809/space-and-time-complexity-find-largest-number-in-a-list/1/module-1-fundamentals-of-programming)



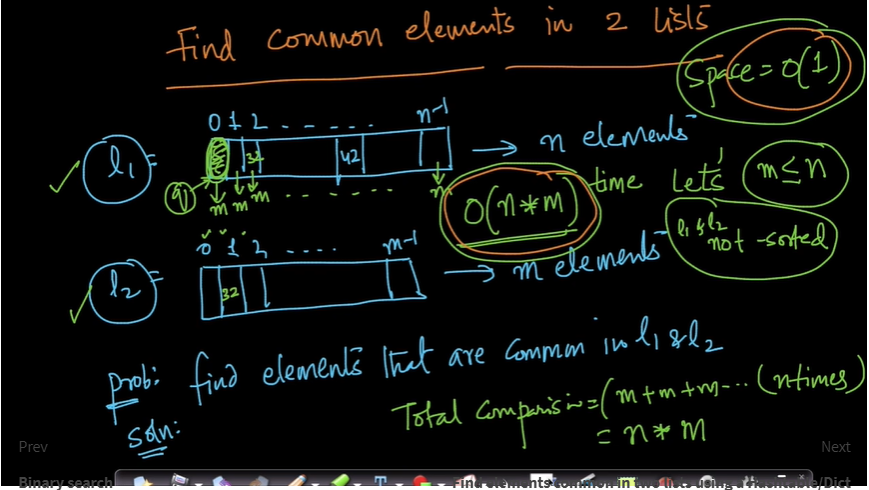


[**Binary search**](https://beta.appliedaicourse.com/lecture/11/applied-machine-learning-online-course/2810/binary-search/1/module-1-fundamentals-of-programming)





[**Find elements common in two lists**](https://beta.appliedaicourse.com/lecture/11/applied-machine-learning-online-course/2811/find-elements-common-in-two-lists/1/module-1-fundamentals-of-programming)



[**Find elements common in two lists using a Hashtable/Dict**](https://beta.appliedaicourse.com/lecture/11/applied-machine-learning-online-course/2812/find-elements-common-in-two-lists-using-a-hashtabledict/1/module-1-fundamentals-of-programming)