Python - Module 8: Strings and string Methods

Python

Module 8: Strings and string Methods

Video: Strings Intro

Welcome to Python sessions.
So in the last class, uh, as part of the,
we discussed about break, continue pass and
before that, like uh, different styles of looping,
different aggregations, how to find
by using looping, we have discussed.
So in this session we are going
to discuss about to string handling.
So how to process the python strings, how
to create the python strings.
What are all the different string methods
these things will see here.
Uh, first how to create a string.
You can create the string,
you can cap the string values in between.
Actually what is a string? First of all,
string is a sequence of characters.
Okay? So here in Python, these strings can be kept in
between single chords or novel chords.
Okay, So let's check one, uh, practical demo.
Let's say as equal to, I'm saying hello word
and one more string.
I'm reading by using double codes also
and integrating the same.
Hello can use single ports

or you can use double codes.
Both result is same.
And you can apply some of the regular
aggregation functions like uh, LAN on the string.
So now it counts number of characters into this string.
Look into this. Uh, part like, uh,
you can apply operators on some string,
actually basically two operators allowed on the string plus
and multiplication.
So multiplication
or asterisk, uh, symbol, which is start for example,
between the string, if you apply the
plus operator, what happens?
Yes. One I'm saying
yes.
Two I'm sayings.
So both the strings, I'm going to conquer net
to con net different strings.
Plus is the operator here.
S equal to S one plus yes. Two.
Check the value of S.
Now both string values got cond here,
Okay?
So between numbers of pluses applied addition will happen
between plus between uh, strings.
If plus is applied, the string of conation will happen.
Similarly, you can apply the surplus in between uh,
A list objects also.
Okay, so depends on object types.
Where you are applying this plus depends on that.
Different functionality will happen.
So technically in object oriented programming,
so this feature is called operator overloading.
Okay? Once we step into the object warranty programming,

we'll discuss about how
to do your own operator power loading.
Okay, so here in between these strings, S one
and S two strings have been perfectly con coordinated.
But between them and I'm expanding a space, then we have
to say like this C SQL to S one plus
after that one space, I'm expecting again plus then S two,
then now print this S two value
print S.
Now between lan you'll find space
and also you can apply multiplication operator also
for the strings.
For example, hello is one string I'm giving? Yes. In. Hello?
If you say yes in to three,
the three times the hello will be repeated.
Check this. Hello. Hello.
So generally this kind of things is used
to construct any underline such kind of things.
So while you suppose if you are formatting a report out,
certain number of lines, like you want
to print some underline, you can use this technique other
here, print,
I'm using hyphen.
Hyphen is a single character multiplied,
multiplied 40, 40 times.
Now same underline is printed here 40 times.
Check the output. Got it.
You can use any symbol just string into string into
that any number that repeats that character,
that many number of times you said 40.
That's why 40 times the hyphen symbol is repeated
here, right?
Similarly print, you can use some star,
star into 40 or 50.

So 50 times the star symbol is repeated.
Got it. So in this way,
automatic operators also can be applied,
but not all the operators, only cent
multiplication operators are allowed.
Addition and multiplication operators are allowed
between the strings
and still a few more things are there relational
operators between the strings.
You can apply all relational operators like I mean
comparison operators between strings.
So this technique we generally use
to compare two different streaks.
So let's execute those things.
For example, S one.
So this is a different uh, cell now come to here. Yes.
One equal to hello
and yes two equal to hello.
Now both values are equal or not.
If I say S one equal to S two, it is showing false
because here small letter H here, uppercase H.
That's why definitely those are not equal.
You got false, okay?
For example, I'm saying that yes one lawyer,
both the things I'm com converting into lower case
after converting them into lower case,
I'm checking their quality.
Now I got but in this way the given two strings are equal
or not to check that the relational operators also can be
applied even on the strings.
You can apply in greater than less than such kind of a
higher layer comparison operators for example,
yes, one is equal to yeah, just two is equal
to B.

Now as for the sequence, a small B is bigger.
Now can you check like uh, yes, one greater than S two
condition falls reverse it, yes.
Two greater than yes.
One condition is
you can apply greater than, greater than
or equal to less than, less than or equal to anything.
You can check it, especially if there are date strings
to compare with the date.
Uh, like to compare date strings,
which which date is earlier, which date is uh, uh,
latter one to compare that also this will be used
for example, I'm taking date one.
Let's say 2022
and March March 25th.
This is one date and the date two
2022 still on the
March and 29.
So that means here, uh, uh, 25th is earliest date,
lateral latest 29.
Okay, so just to compare it, which is uh, lower value,
which is bigger value, it is showing
DT one greater than DT two.
So we did not run the previous sale, not on the same thing.
I got false because 25 is lesser than this 29.
Okay, so check with the less than D one
is less than dated.
D one is less than dated.
Now it is, got it.
In this way you can compare different strengths.
Relational operators, you can apply
the next we see index numbers
generally on the list it couples.
We are applying index numbers to read the values on strings.

Also, you can apply index numbers as you know,
index numbers, basically three types
like positive index numbers, negative index numbers
and negation of index.
So index numbers, arms,
strings in python, I have three types of index numbers.
What are those? Positive
and negative index and one more
negation of index.
Okay, if it is positive index numbers starts from
zero already.
We discussed this part as uh, at the time
of uh, list object.
So again, once again we are discussing the same
start from zero.
The two rates from left to right
and negative index,
it starts from minus one.
It rates from right to to left
negation of index.
This is uh, starting with uh, this tail symbol.
You can use zero. Okay,
but please starts from zero here also
from right to two
Here.
These are the three different
varieties of the index numbers.
Let's uh, start playing with the positive index,
which is zero, starting from zero from left to right.
Look at this word Computer.
Now string value is computer. I want to take post character.
That's all. So it
takes post character.
Similar last character I want

without using native index number
or a index, I want to say this one.
So in competitor, how many words are there?
Totally eight characters are there.
Okay, we now we need not to count it to,
if the index number is starting from zero,
last character index number is seven.
Okay? So for this without competition directly can do that.
Len of yes, minus one len
of ass well is 88 minus one means seven.
So yes, sub seven means the last character,
which is R will be printed.
So these are the, these two are about post minutes.
Similarly, AYA, I want to print this P.
The P is in fourth position, yes of
4, 4, 4.
Fourth one means index number three
because uh, positive index is starting from a zero
right Now let's look into the negative index.
Check the value. The value is not better here.
I want to take lost character, lost the first character.
So negative index, I'm applying by saying minus. Well
that is or similarly lost second character.
Want yourself minus two.
Okay, lost second character. Getting.
Now let's apply the negation of the index.
Negation of index. That is starting with the tool symbol.
Generally if you say a subzero, that is first character.
If you apply the negation symbol, tool symbol for it,
that is last first character.
It'll take it. Okay, last first character.
Similarly second character. Want your soft one?
Generally say if you say til the S soft one last,
the second one, it'll take it now just check it.

We got E. So these are the three types of index numbers
are supported by the Python.
Okay? So in list that you are applying slicing
just to create the subsets.
Subsets from the list.
Similarly, you can create subsets from the given string
by using the slicing.
So our next work will be here
slicing on strings.
So
slicing runs here.
Range of index numbers.
Range of index numbers.
Look into this one. Once again in our string
there is a word called computer.
I want to take first three characters. Yes.
So actually the format is like this.
Your string or list
spot index column.
So this is what the format,
so here the rule is start index is always included
and index is excluded.
For example, I want PUT puts from.
So what is the starting question number?
Uh, p starting question for that means index number three.
I'm giving a soft three column that is starting index
from the third index.
How many characters that you want of PUT?
There are three characters. Okay? Three plus three, six.
So three columns. Six means what?
Index numbers will be selected.
Index number three because that is included
as a starting index.
Number next four, next five. The sixes won't hit.

Six won't be taken. Now just to check it exactly, we get PUT
as part of this expression.
This is what a slicing
similarly faster three characters I want.
I'm saying start next
and index faster.
Three I one simply say three in the end.
Index plus start index. I want from the beginning.
That's why no need to say any start index.
If you say like this, if start index is missed, is skipped,
you'll get from the zero.
That means the faster three characters you are getting.
Similarly, the last three characters, I want
look into the SA or else like last five characters.
I want last five characters are here.
Put I want to get the put now.
Yes, saw last file.
How to get the last two 50 character minus five call.
This time I'm skipping y index.
If I skip an index, it takes still last.
Okay, so all these are the basic operations on the string.
In this session we have L.
So in the next session we start using different
predefined methods on the strings.
Later we'll be learning in the functions concept of how
to create our own functions for anything.
Okay? So for now, like we are going
to use a predefined functions on the string.
We have for different purposes,
different methods are available.
So generally independently,
that is a function when now this is a part
of an object in object oriented programming.
The same functions are called methods.

Yeah, what the strengths? We have different methods, okay?
Which have some predefined functionality.
So those things will,
we are going to learn in the next session.
Thank you very much. See you next time. Bye-bye.

Video: String Methods

Welcome to Penman Sessions.
In the last session we discussed about how
to create the strings and how
to apply operators on the strings
and uh, relational operators on the strings index numbers
and slicing we have seen
and as part of the index numbers, we have seen positive,
negative, and negative of the indexes.
Now in this session we are going to work
with the string methods, especially here, find an index
strip, yellow strip
or strip under case related functions,
lower, upper and tight.
So these things will see,
let's say a string, the title part, our session
string methods spot one,
I'm grading a string.
Now
let's say the line equal
python is simple and is,
so this is a line.
Line is having like a lot of uh, different words.
So one of the sub I will take, for example, I'm taking easy.
I want to find out whether easy is available in
the line or not.

For this, you have an operator
that is called membership operator.
You can say easy in line.
The word easy is available in the given string.
It returns to otherwise it returns false. Okay?
Same thing. Something programming the word programming,
but programming what is not available here.
That's why it s false.
So already as part of the membership operators,
we discussed in not in, let's apply the not in also here
for the programming, not in life,
actually programming not available.
That's why it returns because of not in operator.
So in operator
what it can explain is whether the word available
or not, it can explain it,
but if word available in which position
that is available, it cannot serve.
I want know the position of it.
So then you have to use find or index.
So I will tell you, once I demonstrated these two things,
I will explain what is the difference
between fine and index.
For example, I'm going to check the word easy, okay,
what is our object Do index
or I'm checking with the, the word easy.
Check this. Now the word easy is available.
That is available at 22 index. That means 23rd character.
The starting version of easy is 23rd character.
That means index number 22
because star index is starting from the zero.
Same thing I'm applying with the find dollars,
find dollars, line do five.
Oh please, same output.

You will see Now here in both the cases,
you don't see any difference here.
Now I'm going to show you the difference line index
of for example apple.
The word apple, the word apple is not
available in the given line.
So for the index it returns minus one.
So I'm sorry, you got same thing I'm applying
with the fine.
The line do fine
kind of happen.
Actually apple not available. So that absorb here
the find is return minus one.
So here it can understand the difference.
This is a comment of the find find method.
If given substream
available, it returns
index to index extent
if not available
rate
returns minus one.
So same kind of statement for the index
you've given substream
available with
Start index number same as five
if not available it it throws at
by using exception handling concept, we need to handle that.
Okay?
So programmatically you can operate like this.
Suppose if apple
I'm applying line do kind apple,
the result is minus one, then print
word apple, not a value,
I'm printing something.

A word apple.
It's available at start index.
Find find of that apple.
Now check the message. Actually apple not available.
It is displaying that the word apple not
available in the given stream.
For example, I'm checking the same thing with easy.
The word easy
it is showing that, sorry,
it is a start index of it again.
Once again here you fix it did not change here. Yes.
Now look into this one actually easy available.
That's why it is showing the word happened is available at
start index 22.
In this way conditionally you can operate
by using fine function.
This is not possible by the index function.
So what are the next functions?
We'll see I want
to remove spaces, but these are
how three options, three functions here.
Strip to l, strip or strip.
So firstly work with the L strip
what each function is doing.
Post select right L strip
removes remove left to side spaces
including invisible characters.
Invisible characters for example like a backslash
and backslash to such kind of things.
Backslash and backslash T such things are
called invisible characters.
Okay? Similarly
or stream removes right side
both spaces and invisible characters and a strip.

This removes both sides.
So for this, try to understand the difference.
I'm taking my line. Let's say
Barak three
In both spaces are there,
Okay?
Left side spaces are the right side spaces.
Also there I'm applying line dot, yellow street.
Check the output. It cuts off
only left side spaces.
Still right side spaces are available.
Apply the R strip now line dot r strip.
Now it cuts only right side spaces still left
side spaces are available.
Now I'm applying strip which can cut both left
and right side spaces.
Got it. This is what the strip functionality at the time
of reading data from files.
Very frequently we use these three types of functions
like L stream, L strip, or strip and strip.
Okay, let's check the next function. That is a ch.
That is about changing the cases like a lawyer case,
upper case and there is a title.
So generally the standard for the title is if you want
to apply any title in the document in a word,
the first letter could be in upper case.
Okay, remaining letter to be in the lawyer case.
Let's test with this lawyer. Upper and title
lawyer
words, upper case letters
into lawyer case.
Let's see, the string is hello,
H-E-L-L-W.
I'm typing in a mixer cases. Some letters are in upper case.

Some letters are in lower case.
Lets me keep this uh, H to be in the small other here.
H is in lower case E, upper case second yellow,
lower case second L, upper case w lower case.
I'm saying that yes, not lawyer.
All the things will be converted into lower
case check.
All letters are into lower case.
Similarly upper case, yes, upper,
even lower case characters will be
converted into upper case.
Now I all are in lawyer case.
I want to make it as a title yes, title
here, only first characterize in upper case remaining all
letters, sorry, lower case for example, one more word.
I'm taking let's say line equal to
keeping some in the middle.
Also I'm keeping some upper case letters,
which
there are mixer case letters here.
No maam. Now I'm going to apply the title.
What happens, just second line.
Do title
check for each word.
Actually in the there are two words for each word.
The first letter is in upper case remaining all letters in
layer case, even in the second word serum also first,
first letter is an upper case remaining all in lower case.
So this kind of thing will happen with the titling function.
Okay. As part of this session, we have learned about
majorly eight functions.
Find an index, strict L strip or strip
and lower upper title.
Okay, in the next session we are going to discuss about

different other, other functions like uh,
ease visit is alpha, is alpha numeric.
Okay, ease space
and starts with an end with join split more functions.
We'll see in the next session. Thank you very much.
Let's meet in the next session. Bye-bye.

Video: String Methods part 2

Hello, welcome Python.
So in last session we discussed about some string methods
like uh, find index stripping,
strip with the strip, L strip or strip.
Okay, so in this session,
few more important methods we'll be
discussing as we'll.
Say about ease visit and dissolve.
For example, I'm taking string like this in this,
I'm keeping all the purely numbers, just
what it is connecting in front.
Now I'm testing with the X dot ease visit,
not run it true.
For example, I'm saying that ease alpha,
that means alphabets if the given string
is alphabet or not is testing.
Let's change the values. I'm saying A, B, C,
1, 2, 3, 4, 5, J, B, C.
The both numbers and characters are there.
Now yes, is time testing false? That means it returns true
all cards,
all numbers here.
Okay, now let's remove this.
Okay, before that one more test with the ease alpha.

So ease alpha is a false.
Even the alphabets are there,
but all characters are not alphabets.
So that means when it returns true returns true.
If all pairs outlets are alphabets,
Okay, let's change it with other parts.
And I'm saying Y this
time purely characters.
I'm giving Ys
written
one second.
I'm changing the values on a, B,
C one is a mix of numbers
and also alphabets.
Now in this case extort, he's all nu.
He's all nu means he's all for numeric.
That means it is a combination of
with lum cell characters.
Then it, okay,
so on that way you can verify it.
So he's all nu just you have check right now
and one more function is there
and the given string is that space or not.
If space available, it returns true.
If space not available, it returns false. Okay,
let's check one thing.
I'm saying X equal to balance string.
Now there is no space in the
given string do e space.
It has written me false. Try to use some space there
at least one space.
Now that's the same function E space.
Now it is false.
So still it is showing false

because wait,
that means all together spaces.
Let's test it. Multiple spaces I have here
is space.
Now check the final output.
That means it is ing through
if all gaps, all
pass spaces.
Okay,
let's check the next functions.
So it is all for numeric and use space. We have check rate.
The next functions starts with a specific character
and ends with a specific uh, character.
I'm taking a an example.
Let's say string is employed, right?
I'm checking whether it is starting with EMP and starts with
now.
Finally starting with the letter E, m, P, the set
of Charact, CMP.
That's why it
similarly, whether it is ending with the data
or not, I want to check yes start.
That's the data
other here I'm taking sales data.
Now we start starts with, yeah,
got it.
Okay. Now I have some in a list based on this some uh,
we play something with the list.
So thing that in the list, my file names up there like uh,
the employee dxt, the employee one dxt
that sales do dxt
and sales dot CSV
sales two
list of different files 30

and the products.in different files.
Now here file of Python of I want to filter only LY files.
If it is LY file, have a clue.
The file is starting with EMP. Okay, only LY files.
I'm going to get it now files.
So in Python list they have a comprehension technique.
A format is like this expression or loop in right side.
So This is a format.
On this format we'll apply this EMP files.
Let say here my loop is for fin files.
If it is file the clue that is starting with the starts,
it starts with our E.
So I teach it, the file name is coming into Y
and we are checking with the yahoo dot starts
with if this condition is true, it is ing that file
and only within the EMP files.
Only employer related files to be there. Now check it.
EMP files.
Got it. Only employee files are here.
Similarly, I want to take only sales related files.
Now yes, files equal to same kind of operation.
I'm going to apply on the files.
If you have done starts with starts with our sale.
Now check this. Yes, files
only sales files to be here.
Sales or txt sales. CSV sales. Two txt like this.
I want to take only text files, not the CSV files
and not remaining files.
This time I'm going with
yeah, in files, yeah, yeah.
DOT and suite because I want text files.
Each text file is ending with the do DxD
now into the dxt files
and verify the same thing with adjacent.

Now all these files are EMP, EMP one, dot XT sales,
sales two products.
All these things are text files
in personal scenarios.
Whether it is starting or ending with this one
or not, you can understand same thing.
We are going to separate Cs P files. That's says C files.
You are ending with do Cs.
Now print C files while you see,
right?
These are the examples. Still check other functions.
Now we have seen starts with an end split come
to the giant and split the next level.
These two are also very useful functions.
First I show you split
and then I show you the
giant split.
It splits the text based
on given delimiter.
So
further split method.
Default is a default to deter is space.
That can be single space
or that can be sequence of multiple spaces.
Default to deter is space.
Space or spaces. So let's check this.
I'm taking one employee record.
The record is like this one, one.
His name is Ravi salary, 80,000.
He's a male department level.
And from after here, if you treat this
as an employee regard, there are many entities like id name.
These are the fields, field values.
Each field value is separated with car.

Each field value is separated with comm.
How many commas are here?
Just counted
1, 2, 3, 4, 5.
There are five commas. Five commas means five plus one.
Six fields. If there are end commas, 10 plus one fields.
So here field to field is comma based on the comma.
I want to split them whenever I apply the split.
Split returns list.
Split returns list object.
So here five commas means in the list
you'll get six elements.
Now just observe record.
Let's say the employee list equal two record
dot split splitting based on comm.
Now check E list. Update is a list.
Now it has six elements up in six values.
Got it? Now I want to take the salsal equal to now.
E list of two.
Third one, the two I'm converting that into in
this was split.
So on the pure text world that also you can apply it.
Suppose I have a plain text like this line is equal to
python supports all
programming paradises.
So this is what my text now there are multiple words.
I want to split this line into the words. So word to word.
What is the separatory here? Space is the separator.
So line dots simply say split.
That's a word. Sequel to line dot split.
I'm not giving any delimiter here.
So what is the default limiter?
Default deli space.
Now check the words. What is the words?

Data type is list within that.
Each element is a word python,
one word supports all programming paradigms.
All these things that is there.
This is about the splitting.
Now there is another function, giant function is said.
The purpose of this giant to con coordinate
list elements as a single enemy to con
list elements into, into
single strength.
Let's look into the words, which is about what this, about
what I want to recon what into like string line.
Look at the values of the words once again.
Now each word to be separated with
format is like this deter.
Join off the list for this format.
Okay, now line equals to I want like let's say iPhone
as a deter between word to word as a separat race.
iPhone dot jar. Join off works.
Now just check the line.
But it Python.
iPhone supports iPhone. All iPhone programming.
iPhone paradises like this. I don't want this one.
I just want to say like this line equal to space
as a separator that to single space as the separator
dot join words.
I'll take the value of the line. Now this one.
Okay, now
after this one example I'm taking,
I'm taking a numerical content, which is a hundred,
200, 300, 500.
There are four values I'm trying to join
with these things based on some comma that
as a single string line, I want to convert value

to value the is comma to, pardon me?
I want to form it as a string. Don't join off.
Yes, you will get yeah that
because we now we want
to coordinate string elements means each
element should be a string.
Okay? But here each element of the list is number.
The reason join needs
each element the list
as string, string data.
Now for this, let's come out each value into the
string, the still, or of V
or V in X.
So X has has four values at each values coming into V,
you're applying string function.
For that V, the numeric data will be converted
into string data plan.
Now look at the latest values of the X.
Now each value is a string log.
Now I'm going to form it as a line.
I want ter between each value dot join the X.
Now we don't get any error. Now check the value line.
You got it? Okay.
So ham comm 200 copies center thing is as a single string.
Understood. So this is
what like a string methods were here.
So in the next uh session, we'll discuss more advanced tasks
like a sorting, sorting string already join.
You have seen for this sorter
content, how to apply the join.
Okay? And a string formatting such kind of things.
We'll be landing. Okay? Let's go to the next session.
Thank you very much. Bye-bye.

Video: String Methods part 3

Hello, welcome to Python sessions.
So today we see different tasks based on the strings,
especially sorting like, uh,
displaying data in a formatted way.
Uh, for example, at the signup of uh,
signing up off into the websites, Hmm, we are, uh, giving
password and confirm the password, how to validate that,
that kind of operations.
We'll see The first, let's work on the sorting.
So for this here, the sorted is a function.
So the sorted function can be applied on the numerical data
and also on the data.
For example, I'm taking one string
here as SQL to bar.
I want to solve this sort off.
Yes, it sort the data in a selling order,
but the list of characters, it is given
just the output of sorted.
So is a small letter.
So from small case to from small letter to two.
Big letter in the alphabetically, yes. Sending order.
So yay, A-B-H-R-T like that it has arranged to the data.
So the written data type is listed on the
teacher element character.
I want to ate all these things as a single.
So already has in the previous session has in the giant.
Now I don't want any dealing adjuster.
I want to con coordinate all the independent characters.
That's why it's a blank space.
I have kept, I did not give even space a blank space.
I have join off
that list, sort it off.

Yes. Now check the output of it.
A, A, B, H. That is as a single.
Same thing if you wanted to do into the river order.
That means a descending order.
Then I have to say sort, yes,
sequel, but default.
Sequel to false. False means descending order.
If it is reverse is two that is in descending order.
Check the output of it.
Now D came first. D-R-H-B-E-A.
Okay, so in descending
or alphabetically in descending order,
it has given the list of characters.
I want to con coordinate that dot
join off sort it sorted off
as but reverse equal to.
So this is our uh, first option.
Next I'm going to display the data in the format.
Okay? Format of those strings is like this.
Formatting of the display,
formatting display output.
I'm taking on variable name.
That means let's say
H I'm going to print it.
So generally without formatting, the style is like this.
My name is after that comma, then name
after that, I am
after that, yes, value I want,
but again, like this, you have to print it.
So this is uh, like when our string is upgrade,
you are keeping the string into the coats after that comma
and after that variable, after that comma again,
and then string again like this, it is really tough.
Now the formatted way, the same thing I want

to print in a formatted way.
Print. Yeah, the formatting of the strings
the coats have started with.
Yeah. So within this here, my name is,
I want name value here.
Whenever I want name, value, variable value, keep, uh,
keep the value into, keep the variable into ly brackets.
After that dot I want, and
after that I, um, immediately is value.
I want H yes,
word like this.
Now usually you can format the expected output.
So here, output. My name is ra, I'm 47 years word. Got it.
So this is what like a formatting of the strengths.
Next chapter, one more task here. Confirming the password.
So you have set the time off,
sign up into the any application
and website of web applications you are giving your
user ID and password.
So password. So first time you give one password immediately
it'll ask you to confirm your password,
but it'll accept that.
Otherwise it'll say yeah,
that means the password not matched.
I'm going to give that one. The second task here,
password equal, that's a key two.
Success is my password
and I'm confirming that password.
Key two, success.
Now both the things you just check it.
Password is equal to confirm if both
values are equal to otherwise false, it returns false means.
So in the password, the KK small letter,
but in the confirm the K is upper case letter.

Now try to give the correct things
I'm running in the next cell.
The first letter I'm keeping in capital letter,
upper case letter, the second case, also upper case letters.
Now check both the things.
Now it is okay, how already have seen equality operator
in the strengths in our, in our previous sessions.
Okay, just we are performing the task.
Now we into the next task.
Credit card details.
So of course you have your own credit card details,
but credit card, uh, details to not to be displayed, uh,
completed to the public at the time you are going
to mask it, okay?
Generally your credit card contains is 16 numbers.
So at the time of masking,
accept last four characters remaining,
all the things will be mask.
So how to do that, that operation,
I'm giving a value here.
That's a card number.
I'm giving the card number.
5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 60
total.
Just confirm the level but uh, characters here.
I have 16, but I want
to mask this except last four characters revenue,
all the things I want to mask.
The masking format is like this.
Mos equal to xx iPhone
after that again four iPhone X, iPhone
and I want to display only last four characters of the fix.
This is the mask of the country.
Now card equal to

plus time giving mask plus
between those strings pluses applied
string conation will have.
Okay? Hmm.
Last four characters
of the card number I want to take already.
You know the string supports index numbers,
negative index numbers and also slicing
by taking a negative slicing.
We use this one last four characters. I'm going to pitch it.
Card number minus four
column I have given the sorting number.
Minus four indicates last four character,
last four character here to I did not give any index
or until last it'll take it.
Okay, now just display the content of the card.
Now, now mask wear.
It is showing you the output. Got it.
So this is what like, uh, as part
of this session, our learning.
So mainly the summary,
what you have learned in this session.
One is, uh, like a sorting in a
sending order, descending order.
We have seen in our practical part and displaying the text
and along with the variable values in a formatted way.
After that, confirming password depends,
verifying both strings are equal different.
And finally, how to mask your core details.
So these examples we highlighted in the next example,
in the next session, we see more on
different tasks like a domain name and email violation.
Okay? And, uh,
converting string into the pal drums, pal drums.

Pal drums means, uh, if you reverse
that string still same value that is called uh, pal room.
How to create your own pal Drums, how
to check whether the given strings are a pal, not how to do
that will be see and uh, in a day, in a date.
String value. How
to extract the required things like a month, year and a day.
And gram strings are there.
Anagram strings are, uh, mostly used in my,
in our NLP things natural language process.
If you observe this, uh, listen silent.
These are the two separate words,
but, uh, what are the with what characters that listen,
the word listen is formed
with the same characters without using any extra character.
That by using same set of the characters.
Another string is formed called silent.
Such kind of words are called anagram strings.
Okay? Whether given string is anagram
or not, like, uh, we need to check it.
How to do that check will will be and rearranging the cases.
Rearranging the cases means like, uh,
lower letters into upper case, upper layer,
upper letters into layer case.
So that kind of activity is called rearranging the case.
And in a given, uh, uh, comment
or text, there are some punctuation characters out there,
especially looking to the first line, like a text.
Hello Bar, how are you?
So after how there is a comment bar,
there is some thing explanatory.
Mark and opera, how are you? There is a question mark.
I want to remove those punctuation related characters.
I want to form a, a new text from that. Okay?

So such operations we will, we are going
to learn into the next session.
Okay? See you there. Bye-bye. Thank you.

Video: String Methods part 4

Oh, welcome to Python sessions.
In our last sessions based on the thingss,
we performed some tasks.
So a few more tasks we'll see in this session.
Look into the first example here.
There is some email I, okay, from this email idea, I want
to extract the domain, okay?
Think that the domain is with organization
and then.com, okay?
Uh, for example here, organized name is example example.com.
I'm saying, okay, I want to extract the content from
let's say email equal.
I'm giving@therightexample.com.
So I want to separate this domain,
then domain equal to first.
You split it step by faster by step
splitting into the words list of words, email, dot split
split off splitting based on at the red symbol.
So before at the red there is a name after at the red.
That is now how many at the red symbols are here? Only one.
That's why it is splitted into two words, two elements.
Now check the values of this. Uh, email that words
and domain.
The required thing is, domain is in second.
One domain equals
words of one or minus one.
Now look at the domain.

Similarly, from this domain, I want to take, I want
to extract only the organization.
Now, once again, split it.
SAW is equal to domain dot split
this time splitting based on dot.
Now look at the value of the dot.
How many dots are there in the given text? Only one dot.
That's why. Dot left side one word
and dot right side one word.
Now example is organization name.com. Okay?
That is website extension.
Now I want to take the organization name, watch equal to WC.
Now look at the value of watch
you are getting organization name right?
Come to the next task.
Converting string into pal drums.
Okay, so that means the pal drum generation,
what exactly the meaning of pal room is.
For example, I have a content of there is
reverse this next
V, yay.
K-A-T-A-K-I-B.
Okay, so in this is okay,
but in English, this is not perfect.
Let's say YZ Aja, if you reverse this,
type in a post the right to left postier Z
and got it.
So this is what like uh, in four, same in river sort.
Also same value things are called
just for improvement.
These kind of examples that used, for example,
my string is yes, yes is A, B, C.
I'm going to add something. S soft
minus one, just check what happens.

C, B, M, right? Got it.
ABC is a, ABC is in power order. CV is in order here.
What exactly here, double minus one is doing is jumping.
So jumping in the river, product positive is taking minus
one means a C minus one.
One more minus one is added. That is becoming minus two.
Pop that minus two. One more minus one is added.
That is so minus one, minus two, minus three.
Then it is taking C, B, A, like this. Got it?
So I'm going to Concord.
This pal is
yes plus yes off I call minus one.
You'll get a pal stringer bring
that value in both orders.
You just see A, B, C, C, B,
a reverse reverse it also A, C, C, B, A.
Okay? Now this A, C, CB is a.
Now any value, the given value is
or not, let's check it how to get that in a reverse order.
For example, the given reverse
or let's say a sequel to
I'm taking Azure.
Reverse string Yes of
column, column minus.
Now check if yes equals to S
print given two
strings given string in a formatted way.
We'll write it already has in the formatted way,
given string, yes is eh,
for example, I work condition falls.
There is not pal, right?
Four given string.
It is not given string. Yes.
That value I'm keeping

is not,
yeah, the variable I gave.
So given string is actually his idea.
River, it still the values is his idea.
It both forward string and river string.
Both values are same.
We are printing, you're deciding whether
it is a pal or or not.
Now is a pal.
Let's test with now reverse.
There is not palant.
So this is what our next task here
and
more task will perform.
Look into this one. There is a date string. Okay?
17th or zero four means April. So first there is a string.
The string has the three date components like a day,
iPhone month, iPhone eight.
From that required things I want to extract, I want
to separate the date components.
That second date
in yy.
Yeah ma'am did the order I'm going
to take suppose now this is 2025.
After that month zero four after that date.
Today's date is uh, let's say 17th.
Okay? This is the date I want to, from this date string.
I want to separate this year path month part
and dev part that two, I want them as a numeric data.
I want to extract the numbers from that. Okay?
So in this case, just to split it,
split splitting based on this time iPhone,
whenever I split it, there are two iPhones.
You'll get a list. List contains three values now,

but each value is a three.
Just confirm this,
okay?
Now I want to extract the year, year equal to
W zero.
That is the first one that you do.
Similarly, I want to extract month
ymo and equal to in W.
There is a second one after day. I want a day equal to in W.
Last one so that you can say minus one.
Now all required things are support, uh, separated.
Now prints year,
month
required.
Things are separated. Got it.
So this is what our next task
look into ana anagram strings.
Listen, what characters are involved here?
IE yes, T, L, l.
Same thing with the same character set.
The second characters are the second string is also now,
but what billings are same.
So now if two words formula by same set of characters,
that is called anagram anagram string.
Now given string is anagram or not, I want to check it.
Let's keep something like, uh, I'm taking user input.
Let's say S-T-R-S-T-R is a is a reward
input type your strip.
I'm keeping it to yes, what other
characters has typed always better practice.
There might be spaces I'm removing the spaces
of both left and right side.
In our previous examples, we have seen strip function,
yellow strip power strip strict.

Okay? And
after that I'm converting them into lawyer like this.
Everything is a string. Now, so this is string.
I'm converting. So like, uh, I want
to sort this one in a sending order.
Okay, let's say
this is string one doing the same thing.
Type your string one
and S two input off
type your second string,
same operation I'm doing here string and then lower.
Okay, so let's say yes,
four one, sort it out a room.
I sorted one.
I'm going to sort the first one simply the function sorted,
sorted off as one in ascending order sorted.
Two here, sorted off
two here.
I did not say any reverse option.
So that default is ascending order. Okay?
And then my next work I if
sorted one equals
to two both values then
given string is anagram
given strengths or anagrams.
Yes, print
given strings are
not grams.
Now let's run this. It is asking you to type the string one.
I'm typing the string one fast forward is a listen.
Then next word, what is the next word? Listen.
And silent. Silent.
So what is the confirmation message given sir Anana?
Okay, let's run one more thing.

Lesson.
Second word is lesson.
These two words are not anana, okay?
Just check. What is this? Uh, yes.
One yes, one is here.
Yes, two is less. Let's keep S two.
I'm modifying this S two, S two as silent.
Okay? Yes. One is LS two is silent.
Now both sorted apply, let's say sorted one equal
to sorted off.
S one similarly sorted to equal
to sorted off as two.
Now check both values. What you, what you get, you can see
print sorted one and print sorted.
All characters are ing has sending order.
E, i, l and ST. Same thing for the second house.
That's why when you compare it, it says that both are equal.
Now sorted one equals to sorted because of true.
Previously we got output as another.
If the values are different, you'll get not another. Got it.
This is what the previous task bot we did.
And one more thing. Arranging the cases, rear the cases
here, there is a special function in Python for the strings.
That is swap, swap case already.
Case related functions we have seen like a lawyer,
upper title, such kind of functions.
What lawyer is doing, converting all into lawyer case.
Upper is connoting all into upper case title.
First letter into upper case remaining all into lower case.
But the swap case is different.
Lawyer will be connoted as upper,
upper will be connoted as lawyer.
Okay? That is rearranging the task.
Let's do the practical part.

I'm saying s equal to hello python.
Now here HP or in upper case remaining all in lower case.
Now I want to reverse that.
Simply can say yes dot swap case.
Now check the output, capital hedge will become small hedge,
smaller hedge remaining, all, all into layer.
Now those into upper, this is called swap case.
Let see the final task of this session.
I want to remove punctuations.
So to do these punctuations, we have a, a special
model here called string.
The string related special functions are used here. Okay?
First time importing the module import string.
Let's run it now string is imported.
Let's say now, string dot punctuations
check this what all the different punctuation
characters that we can see.
So these are the different punctuation characters,
which are special characters.
So, but in the given text there can
be a punctuation characters.
I want to remove those punctuation characters.
That is what the task, okay?
This here, there is a text. Hello. Ha, how are you?
So what are the punctuation characters here?
There is a comma, there is a explanatory mark.
And also there is a question. Got it?
There is a punctuation characters.
I want to remove that one. I want, uh,
my expected output is like a play plain text.
Like hello Barath, Hawaii. Got it. So observe this.
We do suppose a comment line is like this
line equal to hello
or, or

this is a thing.
Got it? Now I want to check whether this, uh,
whether the given charact is, uh,
whether given characteristics part of
that punctuation or not.
Just check it. I'm saying comma, whether this comma in
string dot punctuation
comma is a part of punctuation.
If yes, that true, for example, I'm keeping h,
h is not a punctuational character is alphabet.
It returns false. This technique is enough for us.
I'm performing a loop. Left side
expression, right side for loop.
The pan loop is like this for C in line.
Each character I'm reading here, just check
what happens here in the place of, uh, expression.
I'm keeping that C. Now I got a list.
Each characterizes separate element. Okay?
So, but I don't want all the things.
If the word, if the character is not
punctuation, then only I want right?
Said I'm giving if condition. What is that?
If condition, if that C not in
string punctuations,
uh, in the first, in the first, the C value is h,
h is not available in punctuations.
So that this condition is true.
If this entire condition is true,
then only it returns that value.
Now check it here. The sixth element is actually comma.
Okay? Now the punctuation will be eliminated.
Now karma got eliminated.
After explanatory symbol will be eliminated
and after how are you?

Question mark will be eliminated. Eliminated.
All the things have been eliminated.
So this entire thing I want to con coordinate
with a blank ter dot.
Join all this list.
Now, no pun,
look into this one, no pun.
Now the given text is this one
where we have typed, hello, how are you?
Within this punctuation, characters are there,
but in the final output we eliminated those punctuations.
Hello ra, how are you?
All these other different styles
of hosting string operations in our upcoming uh, lessons.
We are going to use all these different techniques in our
different applications and different application programs.
Okay, let's meet in the next
session for the next lower topic.
Thank you very much. Bye.