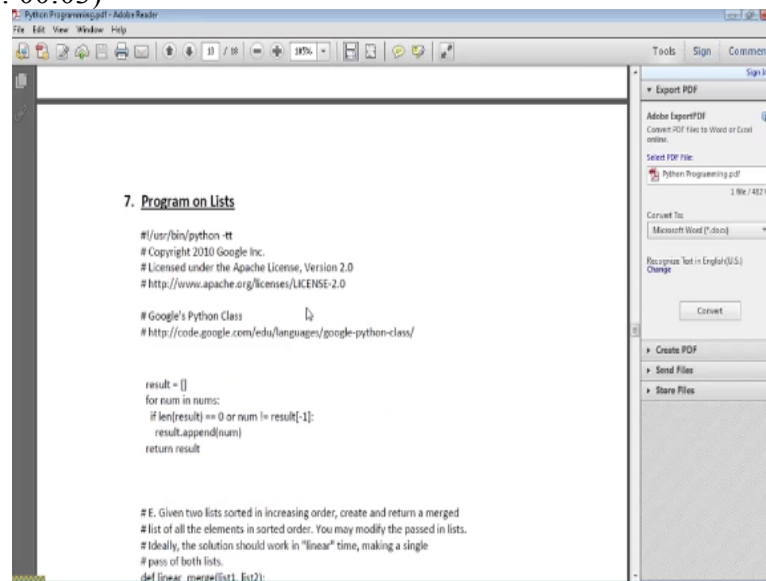


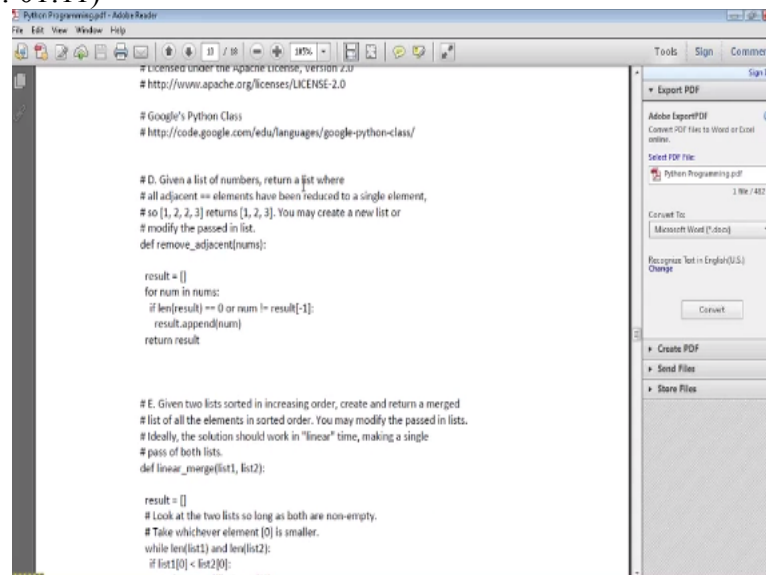
(Refer Slide Time: 00:03)



Hi everyone once again welcome to the LPS course this lecture we are continuing to look at the Python programs the cover six of them today we are going to cover the remaining ones so I am going to go quickly over these things basically so that you get an idea of how the Python code is written and what are the benefits of them and again Python is used for pretty much like. I mean it is a scripting.

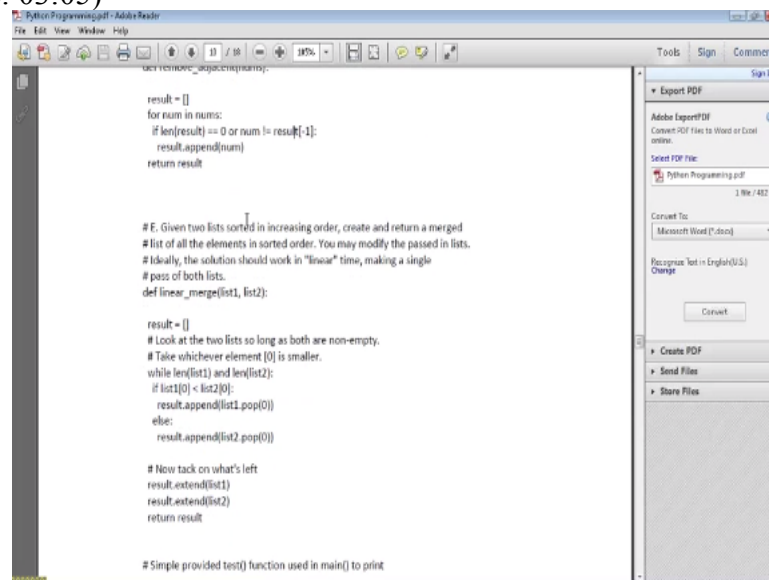
But became them you get the full power of a programming language so here we are actually like going to the number seven which is the programs on lists essentially.

(Refer Slide Time: 01:11)



So here the first one is a simple function given a list of numbers we return a list they are all the adjacent elements have been reduced to a single in I just sent equivalent elements so if it is one to two three the program returns 1 2 3 as a list, so here we define this function remove adjacent

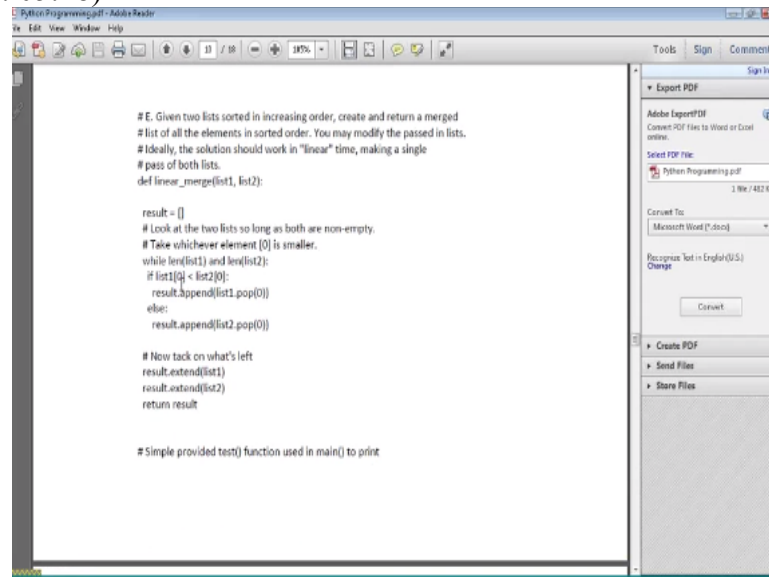
and then basically it is a list of numbers and in the result is also a list or we define as an impious initially and then we basically say core mean you for each of these elements inside those nuns none nums now if length result equal to 0 or num is not equal to a result minus 1 then we append result- happen result we use this particular method that is append with the number. So here you can see basically length inform the length of result is 0 that means that it is a starting point this right now is 0 and then if the number is not equal to the result- result minus 1 that means that the last element then, we append it otherwise we do not have been discontinued on boom until all these numbers are exhausted and then we return the result as the list. So simple program just look at the way that it is written how elegant it is all we do is essentially like in the append that is what we are using it.
(Refer Slide Time: 03:05)



So another one is basically given two lists started in increasing order create and return and merge this of all elements in sorted order you may modify the past innless ideally the solution should work in linear time so what that means is basically or every element it needs to take a fixed amount of time, so that it is not an exponential going big names like that so we make we need to do only one path of our both the lists. So here we define the same function linear log this one on this cool those are the two input ones that we are giving here and then we define a new one called the result list and that is defined as an empty list front, so then now we are going to do basically or while or we look at the tool list so long as both are non and d take whichever element is the smaller so that that is the whole overall algorithm there is these comments you can see one. So while these two lists are non not zero non zero now what we do is basically like this 10 is less than list 20 then we append as list one basically and then what we do is basically like I mean

boom that that is actually removed from the list the third one gets upended that gives removed from the list ,and if it is the other one then we append the be append the second one and then remove that from the list now we track what is left by using result dot x 10 list one and same list too and then we basically in return the list or the result itself again.

(Refer Slide Time: 05:28)



```

# E. Given two lists sorted in increasing order, create and return a merged
# list of all the elements in sorted order. You may modify the passed in lists.
# Ideally, the solution should work in "linear" time, making a single
# pass of both lists.
def linear_merge(list1, list2):

    result = []
    # Look at the two lists so long as both are non-empty.
    # Take whichever element [0] is smaller.
    while len(list1) and len(list2):
        if list1[0] < list2[0]:
            result.append(list1.pop(0))
        else:
            result.append(list2.pop(0))

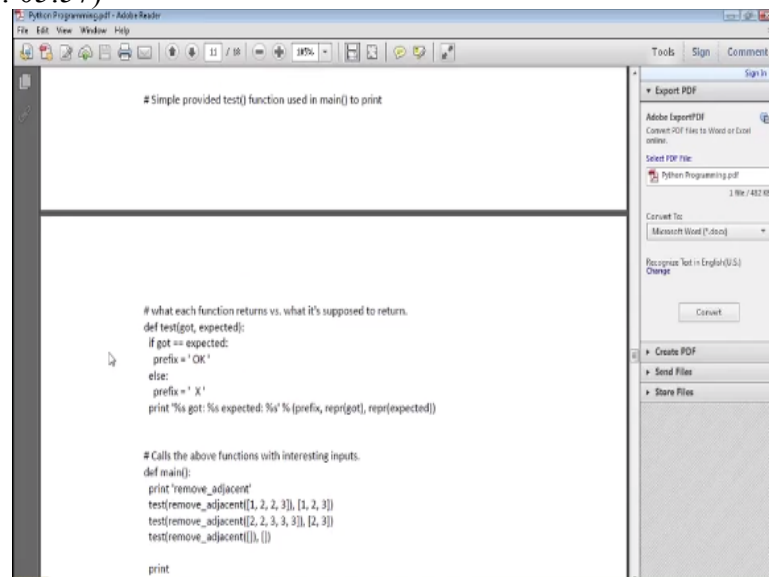
    # Now tack on what's left
    result.extend(list1)
    result.extend(list2)
    return result

# Simple provided test() function used in main() to print

```

It is a fairly simple program to write just keep in mind about this here what we do is basically like whichever one is model we put it in the you know result list then pop it out of the original list.

(Refer Slide Time: 05:57)



```

# Simple provided test() function used in main() to print

# what each function returns vs. what it's supposed to return.
def test(got, expected):
    if got == expected:
        prefix = 'OK'
    else:
        prefix = 'X'
    print '%s got: %s expected: %s' % (prefix, repr(got), repr(expected))

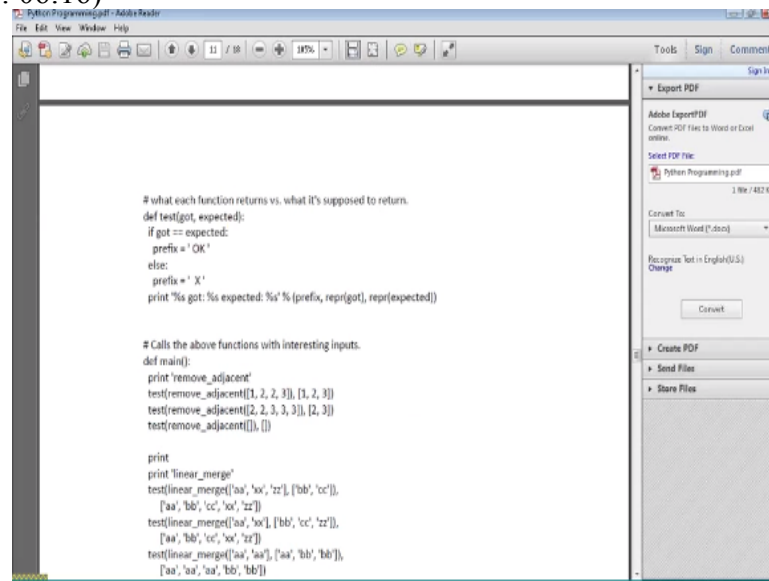
# Calls the above functions with interesting inputs.
def main():
    print 'remove_adjacent'
    test(remove_adjacent([1, 2, 2, 3]), [1, 2, 3])
    test(remove_adjacent([2, 2, 3, 3]), [2, 3])
    test(remove_adjacent([], []))

    print

```

So now a test function using main to print what each function returns was this what it is supposed to return our standard one basically so if not expected then it exists okay, otherwise the six is X and then we print the whole thing.

(Refer Slide Time: 06:16)



```
# what each function returns vs. what it's supposed to return.
def test(got, expected):
    if got == expected:
        prefix = 'OK'
    else:
        prefix = 'X'
    print '%s got: %s expected: %s' % (prefix, repr(got), repr(expected))

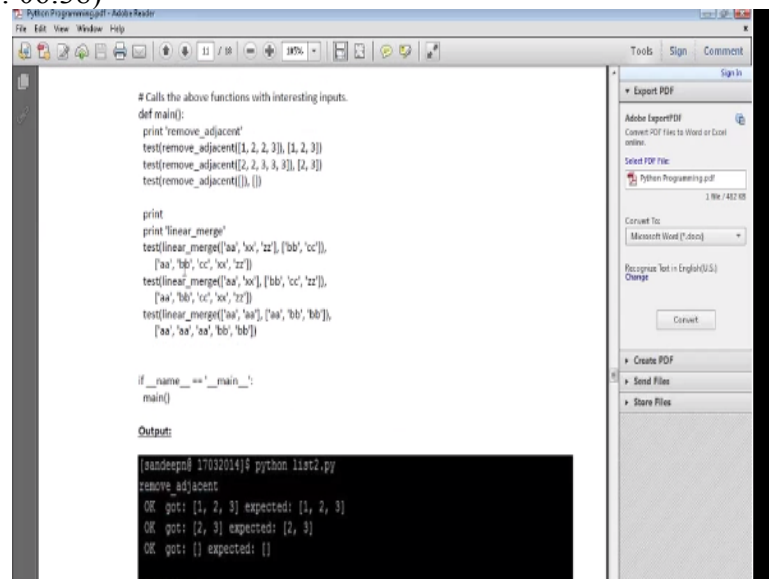
# Calls the above functions with interesting inputs.
def main():
    print 'remove_adjacent'
    test(remove_adjacent([1, 2, 2, 3]), [1, 2, 3])
    test(remove_adjacent([2, 2, 3, 3]), [2, 3])
    test(remove_adjacent([], []), [])

    print
    print 'linear_merge'
    test(linear_merge(['aa', 'xx', 'zz'], ['bb', 'cc']),
         ['aa', 'bb', 'cc', 'xx', 'zz'])
    test(linear_merge(['aa', 'xx'], ['bb', 'cc', 'zz']),
         ['aa', 'bb', 'cc', 'xx', 'zz'])
    test(linear_merge(['aa', 'aa'], ['aa', 'bb', 'bb']),
         ['aa', 'aa', 'aa', 'bb', 'bb'])

if __name__ == '__main__':
    main()
```

So now we test all these programs basically so the remove our descent we just give like 12 23 or and then the expected result 12 23 and then we also try it with some other things basically 22 333 should be this 23 and then if it is an empty list.

(Refer Slide Time: 06:38)



```
# Calls the above functions with interesting inputs.
def main():
    print 'remove_adjacent'
    test(remove_adjacent([1, 2, 2, 3]), [1, 2, 3])
    test(remove_adjacent([2, 2, 3, 3]), [2, 3])
    test(remove_adjacent([], []), [])

    print
    print 'linear_merge'
    test(linear_merge(['aa', 'xx', 'zz'], ['bb', 'cc']),
         ['aa', 'bb', 'cc', 'xx', 'zz'])
    test(linear_merge(['aa', 'xx'], ['bb', 'cc', 'zz']),
         ['aa', 'bb', 'cc', 'xx', 'zz'])
    test(linear_merge(['aa', 'aa'], ['aa', 'bb', 'bb']),
         ['aa', 'aa', 'aa', 'bb', 'bb'])

if __name__ == '__main__':
    main()
```

Output:

```
[sandeep@17032014] python list2.py
remove_adjacent
OK got: [1, 2, 3] expected: [1, 2, 3]
OK got: [2, 3] expected: [2, 3]
OK got: [] expected: []
```

It should print an empty list and then for the linear much we again do like a couple of conditions one is we do a a XX in BB CC and then the BB think you should be involved that and then the if it is a xx and then BB visible the BB and CC should be middle and then we should be end and then aa and then a a BB BB should be like three AAS and then too.

(Refer Slide Time: 07:09)

```
[ 'aa', 'aa', 'aa', 'bb', 'bb' ]

if __name__ == '__main__':
    main()

Output:

[sandeepm@17032014]$ python list2.py
remove_adjacent
OK got: [1, 2, 3] expected: [1, 2, 3]
OK got: [2, 3] expected: [2, 3]
OK got: [] expected: []

linear_merge
OK got: ['aa', 'bb', 'cc', 'xx', 'zz'] expected: ['aa', 'bb', 'cc', 'xx', 'zz']
OK got: ['aa', 'bb', 'cc', 'xx', 'zz'] expected: ['aa', 'bb', 'cc', 'xx', 'zz']
OK got: ['aa', 'aa', 'aa', 'bb', 'bb'] expected: ['aa', 'aa', 'aa', 'bb', 'bb']
```

So here is what we get when we run this program and we can say that actually it has executed successfully so this is one way to actually test these kind of programs.
(Refer Slide Time: 07:21)

```
OK got: ['aa', 'bb', 'cc', 'xx', 'zz'] expected: ['aa', 'bb', 'cc', 'xx', 'zz']
OK got: ['aa', 'aa', 'aa', 'bb', 'bb'] expected: ['aa', 'aa', 'aa', 'bb', 'bb']

8. Program on String

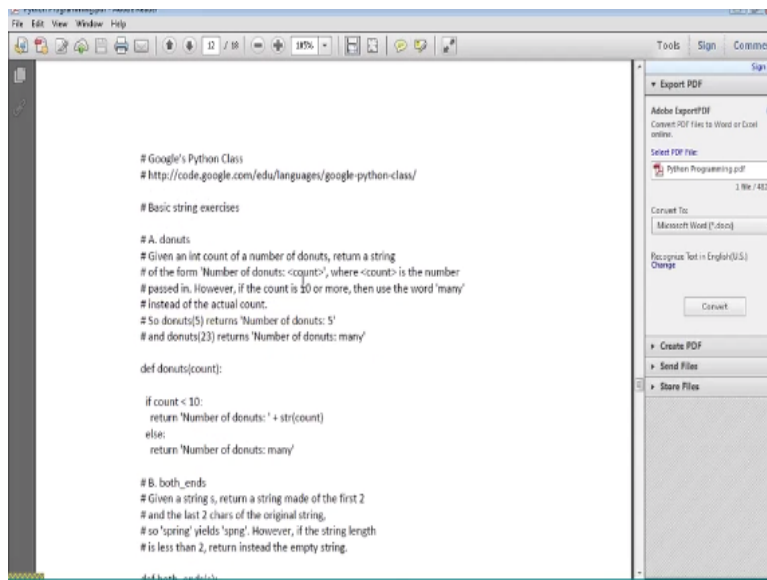
#!/usr/bin/python -tt
# Copyright 2010 Google Inc.
# Licensed under the Apache License, Version 2.0
# http://www.apache.org/licenses/LICENSE-2.0

# Google's Python Class
# http://code.google.com/edu/languages/google-python-class/

# Basic string exercises

# A. donuts
```

I think you should be aware of this and you should be using this kind of testing before you stab it in the echo program string in an assignment.
(Refer Slide Time: 07:37)



```
# Google's Python Class
# http://code.google.com/edu/languages/google-python-class/

# Basic string exercises

# A. donuts
# Given an int count of a number of donuts, return a string
# of the form 'Number of donuts: <count>', where <count> is the number
# passed in. However, if the count is 10 or more, then use the word 'many'
# instead of the actual count.
# So donuts(5) returns 'Number of donuts: 5'
# and donuts(23) returns 'Number of donuts: many'

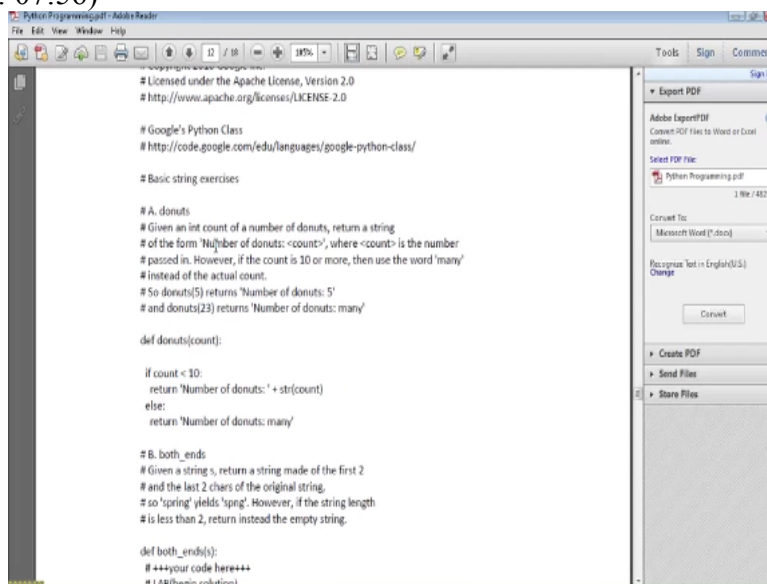
def donuts(count):

    if count < 10:
        return 'Number of donuts: ' + str(count)
    else:
        return 'Number of donuts: many'

# B. both_ends
# Given a string s, return a string made of the first 2
# and the last 2 chars of the original string.
# so 'spring' yields 'sprng'. However, if the string length
# is less than 2, return instead the empty string.
```

Now let us look at some of the programs or our string again we will go through the similar kind of testing procedure so let us see what kind of things that we can do so the number one a sec donuts.

(Refer Slide Time: 07:56)



```
# Licensed under the Apache License, Version 2.0
# http://www.apache.org/licenses/LICENSE-2.0

# Google's Python Class
# http://code.google.com/edu/languages/google-python-class/

# Basic string exercises

# A. donuts
# Given an int count of a number of donuts, return a string
# of the form 'Number of donuts: <count>', where <count> is the number
# passed in. However, if the count is 10 or more, then use the word 'many'
# instead of the actual count.
# So donuts(5) returns 'Number of donuts: 5'
# and donuts(23) returns 'Number of donuts: many'

def donuts(count):

    if count < 10:
        return 'Number of donuts: ' + str(count)
    else:
        return 'Number of donuts: many'

# B. both_ends
# Given a string s, return a string made of the first 2
# and the last 2 chars of the original string.
# so 'spring' yields 'sprng'. However, if the string length
# is less than 2, return instead the empty string.

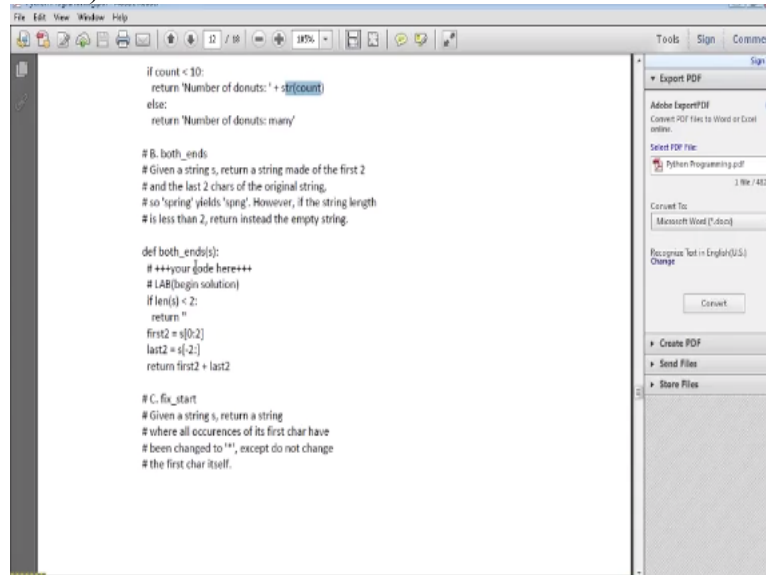
def both_ends(s):
    # +++your code here+++
    # IAR(begin solution)
```

Basically which is even an integer count of a number of donuts return a string in the form of number of donuts The Count account is a number past in our is the number if the count is 10 or more then the use the word meaning instead of the active account so if donuts five returns number of donuts is five but it is going to stay 3 returns number four knocks me so how do we do that ,so here we again define the function called donuts.

And then we pass the counts as input and if the count is less than 10 then we return the number of donuts is plus so number of donuts Colin and then we print this string count it is essentially like whatever number that is less than 10 else.

We just return number of donuts and so basically like this is a very short program to test the I mean to do what we are we wanted.

(Refer Slide Time: 09:10)



```
File Edit View Window Help
if count < 10:
    return 'Number of donuts: ' + str(count)
else:
    return 'Number of donuts: many'

# B. both_ends
# Given a string s, return a string made of the first 2
# and the last 2 chars of the original string.
# so 'spring' yields 'spng'. However, if the string length
# is less than 2, return instead the empty string.

def both_ends(s):
    # +++your code here+++
    # LAB(begin solution)
    if len(s) < 2:
        return ''
    first2 = s[0:2]
    last2 = s[-2:]
    return first2 + last2

# C. fix_start
# Given a string s, return a string
# where all occurrences of its first char have
# been changed to '*', except do not change
# the first char itself.
```

Now the next one is basically both ends in some of the a function you are essentially like I mean if length of this string is less than two then we learned nothing the first two is basically like if the string to the two values and then the last two is actually string - 2 Colin basically that specifies the last two and then we written the first two in the law okay.

So this is basically the it prevents the two characters at both the ends of the string now a fifth start essentially.

(Refer Slide Time: 10:03)


```

File Edit View Window Help
Python Programming.pdf - Adobe Reader
12 / 18 100%
Tools Sign Comment
Sign In
Export PDF
Adobe ExportPDF
Convert PDF files to Word or Excel online.
Select PDF File:
Python Programming.pdf
1 file / 432 KB
Convert To:
Microsoft Word (.docx)
Recognize Text in English (U.S.)
Change
Convert
Create PDF
Send Files
Store Files

# LAB(begin solution)
if len(s) < 2:
    return ""
first2 = s[0:2]
last2 = s[-2:]
return first2 + last2

# C. fix_start
# Given a string s, return a string
# where all occurrences of its first char have
# been changed to '*', except do not change
# the first char itself.

# e.g. 'babble' yields 'ba*le'
# Assume that the string is length 1 or more.
# Hint: s.replace(stra, strb) returns a version of string s
# where all instances of stra have been replaced by strb.

def fix_start(s):
    # +++your code here+++
    # LAB(begin solution)

```

This is given a string S return a string where all occurrences of the first character have been changed to star except do not change the first character itself.
(Refer Slide Time: 10:19)

```

Python Programming.pdf - Adobe Reader
12 / 18 100%
Tools Sign Comment
Sign In
Export PDF
Adobe ExportPDF
Convert PDF files to Word or Excel online.
Select PDF File:
Python Programming.pdf
1 file / 432 KB
Convert To:
Microsoft Word (.docx)
Recognize Text in English (U.S.)
Change
Convert
Create PDF
Send Files
Store Files

# where all occurrences of its first char have
# been changed to '*', except do not change
# the first char itself.

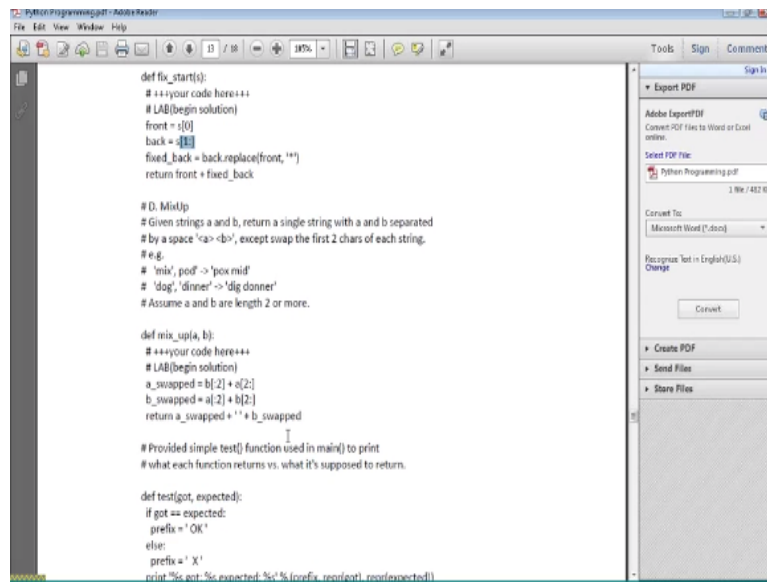
# e.g. 'babble' yields 'ba*le'
# Assume that the string is length 1 or more.
# Hint: s.replace(stra, strb) returns a version of string s
# where all instances of stra have been replaced by strb.

def fix_start(s):
    # +++your code here+++
    # LAB(begin solution)
    front = s[0]
    back = s[1:]
    fixed_back = back.replace(front, '*')
    return front + fixed_back

# D. MixUp
# Given strings a and b, return a single string with a and b separated
# by a space 'a<space>b', except swap the first 2 chars of each string.

```

So if it is a babble then that will become be a star any so the solution is here because we get the string S and then we define what is the front it is 0 and what is the back which is minus 1 and then they say like I mean fixed back which is we use this method back dot V replace in and that we with so this gives you the rest of the character physically one column rest is dissipating all the ones that are after the first one the first one we are we capture in more form and then so once you want to fix that this back basically they replace the front.
This is this character with staff so that replaces all the remaining occurrences of this front director with the staff and then we just returned front and fixed back.
(Refer Slide Time: 11:28)



```
def fix_start(s):
    # +++your code here+++
    # LAB(begin solution)
    front = s[0]
    back = s[1:]
    fixed_back = back.replace(front, '')
    return front + fixed_back

# D. MixUp
# Given strings a and b, return a single string with a and b separated
# by a space 'a> <b>', except swap the first 2 chars of each string.
# e.g.
# 'mix', 'pod' -> 'pox mid'
# 'dog', 'dinner' -> 'dig donner'
# Assume a and b are length 2 or more.

def mix_up(a, b):
    # +++your code here+++
    # LAB(begin solution)
    a_swapped = b[:2] + a[2:]
    b_swapped = a[:2] + b[2:]
    return a_swapped + ' ' + b_swapped

# Provided simple test() function used in main() to print
# what each function returns vs. what it's supposed to return.

def test(got, expected):
    if got == expected:
        prefix = 'OK'
    else:
        prefix = 'X'
    print '%s got: %s expected: %s' % (prefix, repr(got), repr(expected))

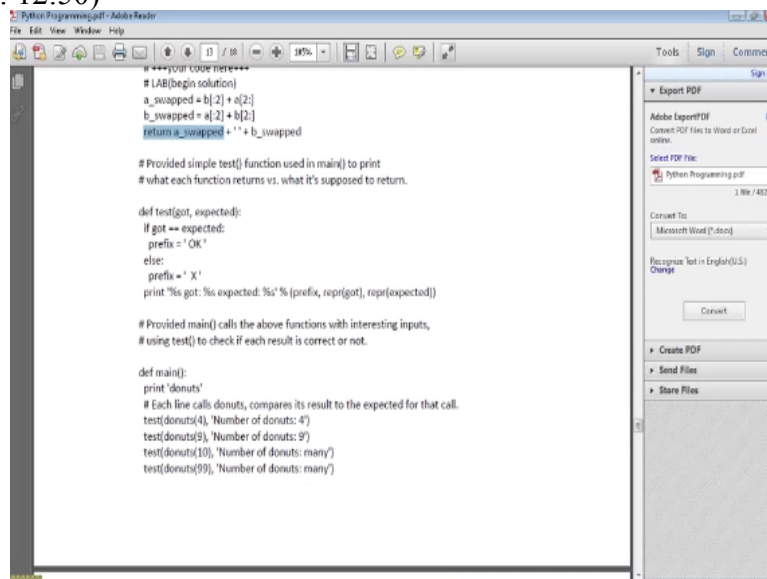
# Provided main() calls the above functions with interesting inputs,
# using test() to check if each result is correct or not.

def main():
    print 'donuts'
    # Each line calls donuts, compares its result to the expected for that call.
    test(donuts(4), 'Number of donuts: 4')
    test(donuts(5), 'Number of donuts: 5')
    test(donuts(10), 'Number of donuts: many')
    test(donuts(99), 'Number of donuts: many')
```

I think these are fairly easy I think you should be able to understand most of the stuff for still go through some of these things, so here the mix-up basically given strings a and b return on a single string with a and b separated by space that is a be except the swap the first two characters of equals so mix part will become Fox mid or dog donner will become dig done so you define again the function boots up a B.

And then essentially like we define the a swap is be the first to column plus a 2 onwards the remaining and then the swap is same thing with yes first two characters and then the remaining of the B and then we simply basically this one return the a swap then space and then this swapped.

(Refer Slide Time: 12:50)



```
# +++your code here+++
# LAB(begin solution)
a_swapped = b[:2] + a[2:]
b_swapped = a[:2] + b[2:]
return a_swapped + ' ' + b_swapped

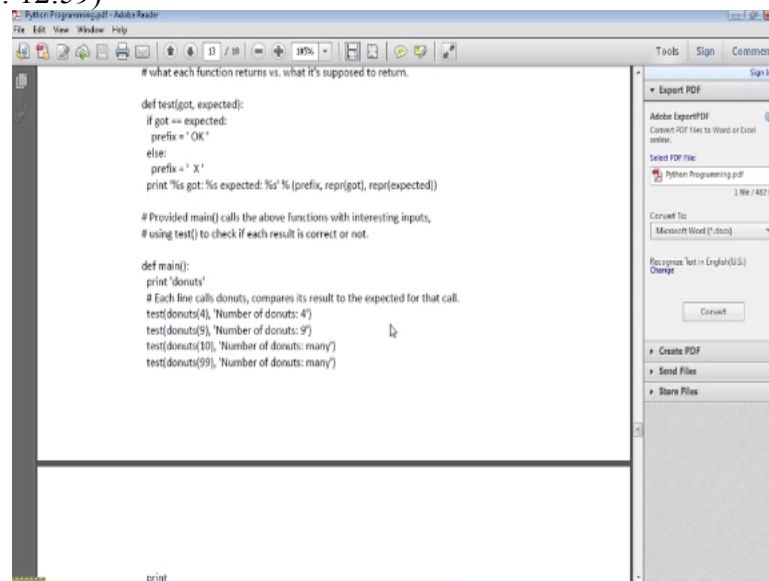
# Provided simple test() function used in main() to print
# what each function returns vs. what it's supposed to return.

def test(got, expected):
    if got == expected:
        prefix = 'OK'
    else:
        prefix = 'X'
    print '%s got: %s expected: %s' % (prefix, repr(got), repr(expected))

# Provided main() calls the above functions with interesting inputs,
# using test() to check if each result is correct or not.

def main():
    print 'donuts'
    # Each line calls donuts, compares its result to the expected for that call.
    test(donuts(4), 'Number of donuts: 4')
    test(donuts(5), 'Number of donuts: 5')
    test(donuts(10), 'Number of donuts: many')
    test(donuts(99), 'Number of donuts: many')
```

So again same function that we defined this basically and then here it is for testing this either a above complete and then we just find the main as basically.
(Refer Slide Time: 12:59)



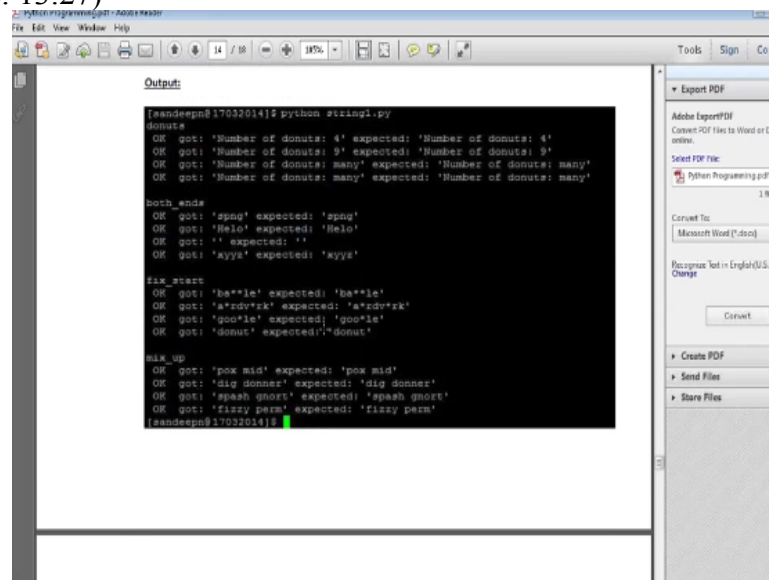
```
# what each function returns vs. what it's supposed to return.

def test(got, expected):
    if got == expected:
        prefix = 'OK'
    else:
        prefix = 'X'
    print '%s got: %s expected: %s' % (prefix, repr(got), repr(expected))

# Provided main() calls the above functions with interesting inputs,
# using test() to check if each result is correct or not.

def main():
    print 'donuts'
    # Each line calls donuts, compares its result to the expected for that call.
    test(donuts(4), 'Number of donuts: 4')
    test(donuts(9), 'Number of donuts: 9')
    test(donuts(10), 'Number of donuts: many')
    test(donuts(99), 'Number of donuts: many')
```

What is the function target what should be the No now the return value you and when we test these different functions within sitting with all the core functions we get polling output which is still do not want they have written the code correct.
(Refer Slide Time: 13:27)



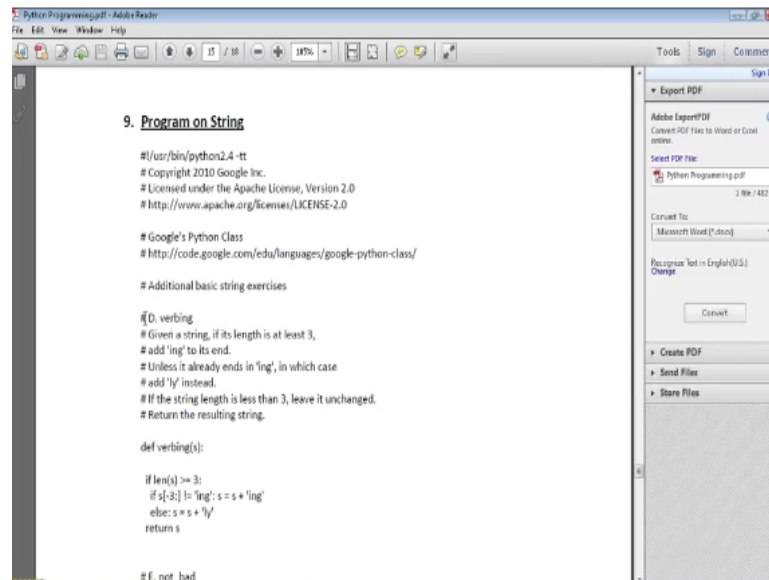
```
Output:
[saandee@17022014] python string1.py
donuts
OK got: 'Number of donuts: 4' expected: 'Number of donuts: 4'
OK got: 'Number of donuts: 9' expected: 'Number of donuts: 9'
OK got: 'Number of donuts: many' expected: 'Number of donuts: many'
OK got: 'Number of donuts: many' expected: 'Number of donuts: many'

Both_ends
OK got: 'spng' expected: 'spng'
OK got: 'Helo' expected: 'Helo'
OK got: '' expected: ''
OK got: 'xyys' expected: 'xyys'

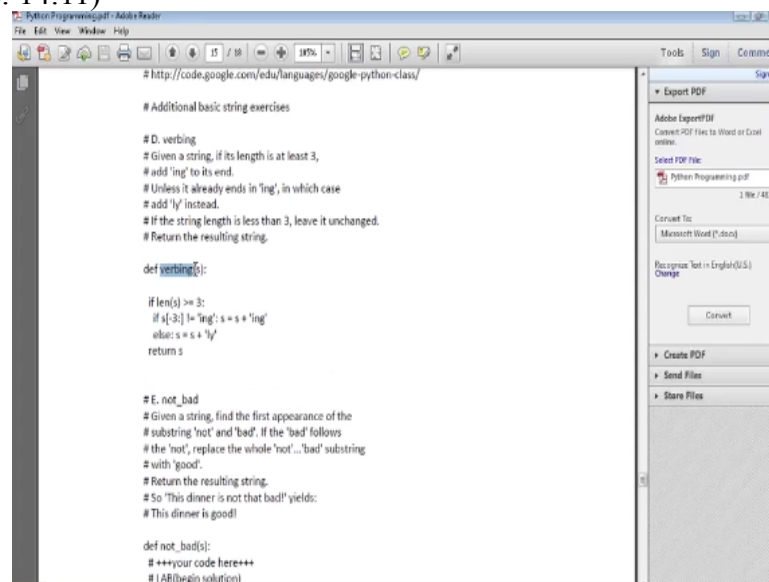
Sim_start
OK got: 'ba*le' expected: 'ba*le'
OK got: 'a*advr'a' expected: 'a*advr*pk'
OK got: 'goo*le' expected: 'goo*le'
OK got: 'donut' expected: 'donut'

Mix_up
OK got: 'paw mid' expected: 'paw mid'
OK got: 'dig dinner' expected: 'dig dinner'
OK got: 'spash gnort' expected: 'spash gnort'
OK got: 'fizzy perm' expected: 'fizzy perm'
[saandee@17022014]
```

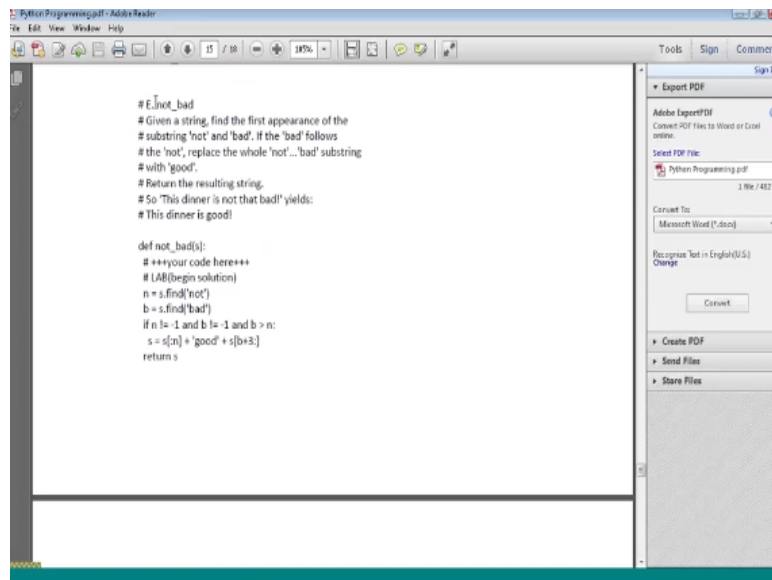
So now more programs on string here.
(Refer Slide Time: 13:40)



Basically this called the verbal this is given a string with its length is at least three add ing to the end unless it is already ends in the engine in which case add ly instead if the string length is less than three we become things and return the resulting screen.
(Refer Slide Time: 14:11)



So there is the our little function which takes s as the input string now we first of all we see whether the link is less than or equal to 3 if that is there sorry if it is greater than or equal to 3 so if it is there then we try to see whether the last 3 characters are ing this is f minus 3 column and if it is not ing then we add the ing to two then but if it is ing its goes into the elf now we add ly has a string and appended to the original string.
And then we return this S so in this case if it is less than three it basically leaves it as is.
(Refer Slide Time: 15:18)



Now the next function is not bad which is given a string find the first appearance of such thing not and bad if bad follows the not then replace the whole not bad sub synced with good return the resulting string , so if it is a basically donner is not that bad then you get donner is good this is so it is not that and that all anything that is in between more and that is basically replaced with this good, so the way that we write it is basically when we get the string then we find not so this is a method and we also find bad store it in these two variables nb.

And then if you if they are not the last characters last once essentially which is in not equal to 1 or be N one and if B is greater than N then we look at the string and then essentially replace form the last term the in character essentially, so we take out the characters up to N and put that good there and then the remaining basically so basically those three.

So that means that essentially we continue after the bad so we go up to dot print good and then go after her the bad so this print the whole new string.

(Refer Slide Time: 17:43)

```
Python Programming.pdf - Adobe Reader
File Edit View Window Help
Tools Sign Comment

# F. front_back
# Consider dividing a string into two halves.
# If the length is even, the front and back halves are the same length.
# If the length is odd, we'll say that the extra char goes in the front half.
# e.g. 'abcde', the front half is 'abc', the back half 'de'.
# Given 2 strings, a and b, return a string of the form
# a-front + b-front + a-back + b-back

def front_back(a, b):
    # +++your code here+++
    # LAB(begin solution)
    # Figure out the middle position of each string.
    a_middle = len(a) / 2
    b_middle = len(b) / 2
    if len(a) % 2 == 1: # add 1 if length is odd
        a_middle = a_middle + 1
    if len(b) % 2 == 1:
        b_middle = b_middle + 1
    return a[a_middle] + b[b_middle] + a[a_middle:] + b[b_middle:]

# Simple provided test() function used in main() to print
# what each function returns vs. what it's supposed to return.
def test(got, expected):
    if got == expected:
        prefix = 'OK'
    else:
        prefix = 'X'
    print '%s got: %s expected: %s' % (prefix, repr(got), repr(expected))

# main() calls the above functions with interesting inputs,
```

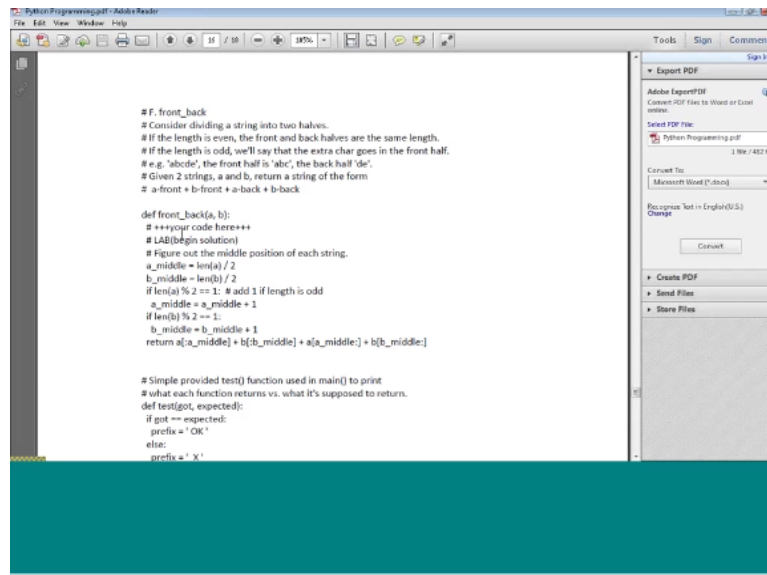
So most of these kinds of algorithms as Del explanation.
(Refer Slide Time: 17:48)

```
Python Programming.pdf - Adobe Reader
File Edit View Window Help
Tools Sign Comment

# F. front_back
# Consider dividing a string into two halves.
# If the length is even, the front and back halves are the same length.
# If the length is odd, we'll say that the extra char goes in the front half.
# e.g. 'abcde', the front half is 'abc', the back half 'de'.
# Given 2 strings, a and b, return a string of the form
# a-front + b-front + a-back + b-back

def front_back(a, b):
    # +++your code here+++
    # LAB(begin solution)
    # Figure out the middle position of each string.
    a_middle = len(a) / 2
    b_middle = len(b) / 2
    if len(a) % 2 == 1: # add 1 if length is odd
        a_middle = a_middle + 1
    if len(b) % 2 == 1:
        b_middle = b_middle + 1
    return a[a_middle] + b[b_middle] + a[a_middle:] + b[b_middle:]
```

I will go through it but I think like I mean once you read it the it will be fairly easy for you on the
phone all these algorithms are returned then as you can see like.
(Refer Slide Time: 18:01)



```
# F. front_half
# Consider dividing a string into two halves.
# If the length is even, the front and back halves are the same length.
# If the length is odd, we'll say that the extra char goes in the front half.
# e.g. 'abcde', the front half is 'abc', the back half 'de'.
# Given 2 strings, a and b, return a string of the form
# a-front + b-front + a-back + b-back

def front_half(a, b):
    # +++your code here+++
    # LAB(begin solution)
    # Figure out the middle position of each string.
    a_middle = len(a) // 2
    b_middle = len(b) // 2
    if len(a) % 2 == 1: # add 1 if length is odd
        a_middle = a_middle + 1
    if len(b) % 2 == 1:
        b_middle = b_middle + 1
    return a[a_middle:] + b[b_middle:] + a[:a_middle] + b[:b_middle]

# Simple provided test() function used in main() to print
# what each function returns vs. what it's supposed to return.
def test(got, expected):
    if got == expected:
        prefix = 'OK'
    else:
        prefix = 'X'

    print('%s: %s got: %s expected: %s' % (prefix, test.__name__, got, expected))

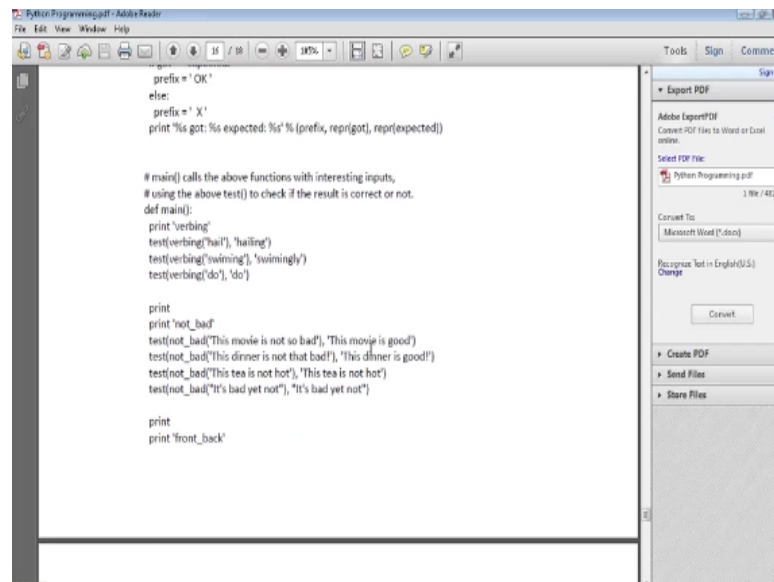
# Call the function and see what happens.
test(front_half('abcde', 'fghij'), 'afbgchidej')
```

I mean less than 10 lines of code, so you should be able to pick it up very easily okay, so the next one is the front back basically which is the dividing a string into two halves the length is even then the front and the back half same then the length is hard they will say extra character goes in the front of that is ABCDE the front half is ABC and the backpack DE is given 2 strings A and B we return a string in the form of A front B front and then A back B back.

So we take these two strings basically and now we need to figure out the middle position of each string so a middle is linked $A/2$ and $B/2$ now we see whether it is an even or an odd if this is odd now add one to the length, one is the length is odd and then so we add that basically in middle plus one if it is if the same thing goes for the be helpful so you see basically like when we $/2$ we do not put the decimal ,so that it is an integer division and it is a lower number so that is idea is adding one.

If you use like dot 0 or something here then have a measure whole population so please do not do that , and then finally we return basically the a string from zero all the way to interval and then be swing with zero all the way to be middle and then a string a middle A the N and then be with all the end.

(Refer Slide Time: 20:32)



So again we do the testing program on them test it and here we see that basically instant .
(Refer Slide Time: 20:47)

```
[sandeepn@17032014]$
```

10. Example Program on Game

```
# Sample Python/Pygame Programs
# http://programarcadegames.com/

# Default high score
high_score = 0

# Try to read the high score from a file
try:
    f = open("high_score.txt", "r")
    high_score = int(f.read())
    f.close()
```

So this is the example program on the game this is probably the last one here first we say basically the high score for the game weevil and now we do a try basically maximum with the I code or text essentially and then, so basically there is already exceptions building input this one looks like that so we try to read the high score from this file that is the whole intent so whether we just read that high core.
(Refer Slide Time: 21:28)


```
# Default high score
high_score = 0

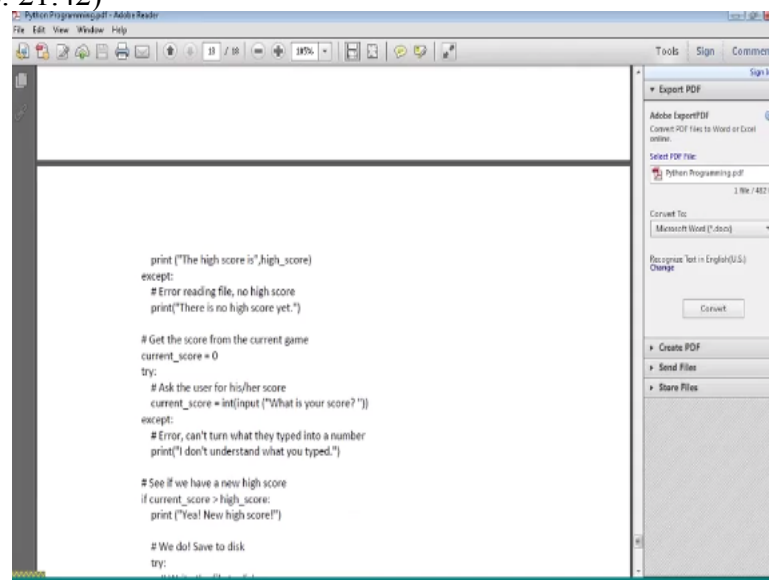
# Try to read the high score from a file
try:
    f = open("high_score.txt", "r")
    high_score = int(f.read())
    f.close()
```

```
print ("The high score is",high_score)
except:
    # Error reading file, no high score
    print("There is no high score yet.")

# Get the score from the current game
current_score = 0
try:
```

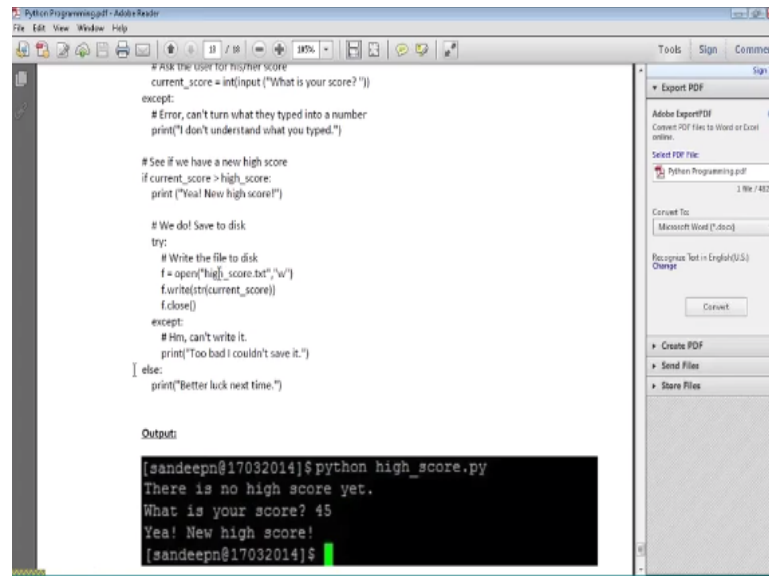
And then we close the file and then we print the high score and then we generate any questions.

(Refer Slide Time: 21:42)



Let me try to get the score from the current game this is current score is zero and then ask the user for the score and then again, we have an exception problem if it is type something else and then if current score is greater than the high score then we say like GIU you got a new score and then we also open the thing.

(Refer Slide Time: 22:11)



And then write out the new score into the file and then if it is not there then they say like print better luck next time , so here this is one thing that of the programmers, so that is pretty much all that I want to cover for today I hope like Python programming is square and enjoyable with these kind of lipids we can actually build bigger programs very easily and I would like you to try those kind of things and these are all like some basic programs, which from which you can that bigger and better profits thank you very much okay bye.