INDIAN INSTITUTE OF TECHNOLOGY MADRAS

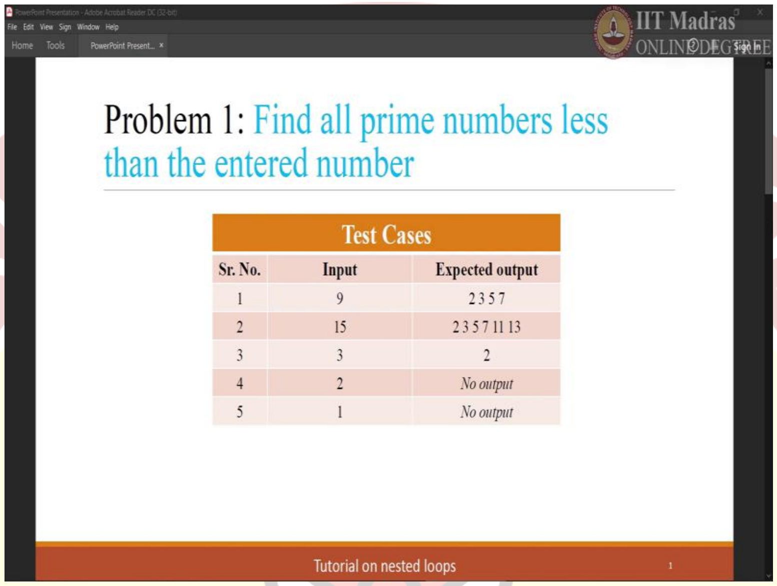
सिद्धिर्भवति कर्मजा

IIT Madras

ONLINE DEGREE

Programming in Python
Professor. Sudarshan Iyengar
Department of Computer Science and Engineering
Indian Institute of Technology, Ropar
Tutorial on nested loops

(Refer Slide Time: 0:16)



Problem 1: Find all prime numbers less than the entered number

Test Cases

Sr. No.	Input	Expected output
1	9	2 3 5 7
2	15	2 3 5 7 11 13
3	3	2
4	2	*No output*
5	1	*No output*

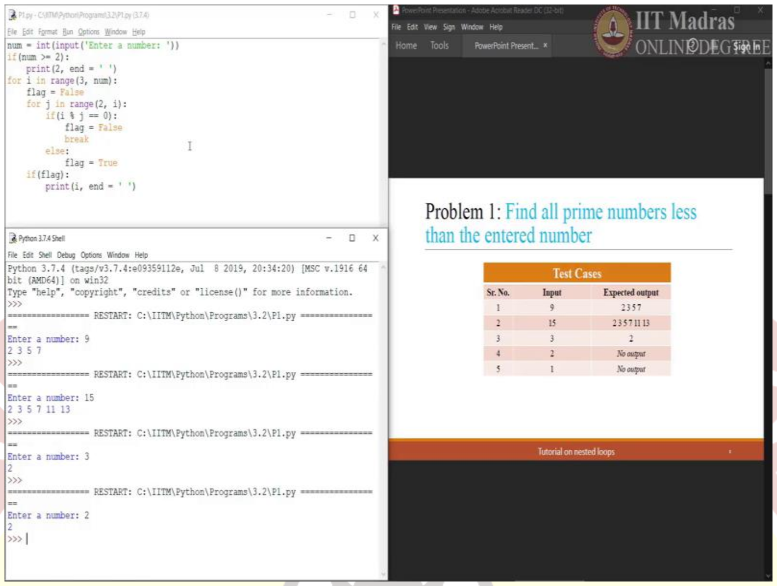
Tutorial on nested loops 1

Hello Python students. In this tutorial we will see nested loops, we have studied two different types of loops, while and for, and nesting can be done using these two loops in any order. As in, for inside for, while inside while, for inside while or while inside for, all four different variations are possible with respect to nested loops. Now, we will see four different examples where these four different types of nesting is used.

Problem statement 1 find all prime numbers less than the entered number, here as per the test cases we have to give one number as input and the Python program should return all the numbers which are lesser than the entered number and at the same time the numbers has to be prime numbers.

For example, if the input number is 9 then the expected output should be 2, 3, 5 and 7. Similarly, for 15, where it should be 2, 3, 5, 7, 11 and 13. For input 3 it should be only 2, whereas for input 2 the Python program should not print any output because there is no prime number which is lesser than 2. Same for last test case where input is 1. Now, let us look at the Python code which represents this problem statement.

(Refer Slide Time: 1:54)



```python
num = int(input('Enter a number: '))
if(num >= 2):
    print(2, end = ' ')
for i in range(3, num):
    flag = False
    for j in range(2, i):
        if(i % j == 0):
            flag = False
            break
        else:
            flag = True
    if(flag):
        print(i, end = ' ')
```

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================ RESTART: C:\IITM\Python\Programs\3.2\P1.py ===============
==
Enter a number: 9
2 3 5 7
>>>
================ RESTART: C:\IITM\Python\Programs\3.2\P1.py ===============
==
Enter a number: 15
2 3 5 7 11 13
>>>
================ RESTART: C:\IITM\Python\Programs\3.2\P1.py ===============
==
Enter a number: 3
2
>>>
================ RESTART: C:\IITM\Python\Programs\3.2\P1.py ===============
==
Enter a number: 2
2
>>>
```

Problem 1: Find all prime numbers less than the entered number

Test Cases		
Sr. No.	Input	Expected output
1	9	2 3 5 7
2	15	2 3 5 7 11 13
3	3	2
4	2	*No output*
5	1	*No output*

Tutorial on nested loops

In this code the first line we will accept the number from user using input function then convert it to integer and store in a variable name num. If the number is greater than equal to 2 we will simply print 2 as we know is the only even number which is prime. Hence, we will have this particular if condition specific to number 2.

If enter number is greater than equal to 2, print 2. And we are using this end space over here so that every time print function will print a number in the same line. Then for numbers 3 onwards we are using this for loop which will iterate from 3 to the entered number. We will set the flag to false.

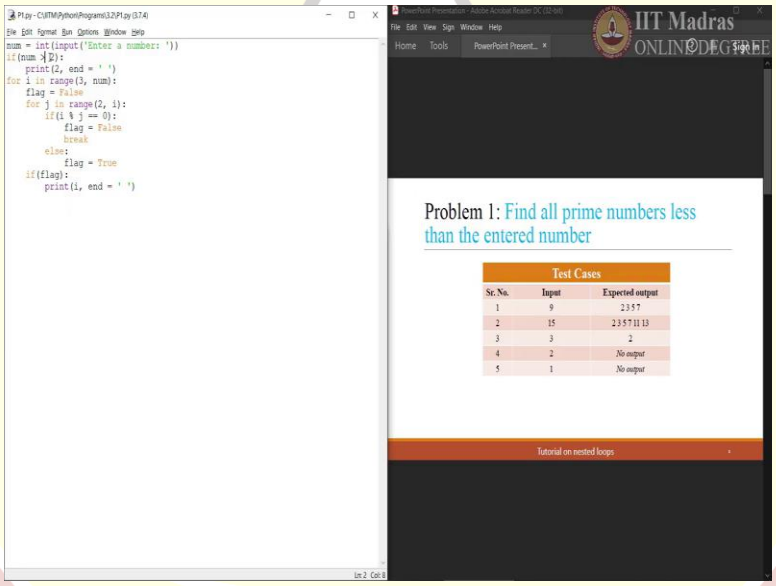
Now, as we want to find all the numbers which are lesser than the entered number we will use this particular i as input for next particular loop. Therefore, the range for inner for loop will be from 2 to i, the outer loop will iterate from 3 to the entered number and each of those numbers will be later on check whether they are prime or not using the inner for loop.

Therefore, we are doing this particular operation i modulo j and if the result is 0 then we are setting flag to false that means there exists at least one number which is j in this case dividing i. Hence, flag will set to false and we will break the loop. This particular break statement will break only the inner loop. Because if this condition satisfies we can term that particular number as not prime. If this flag never becomes false, then we will set it to true saying it is a prime number.

And then at the end outside the loop we will check if flag is true or not. If flag is true we will print that number as prime once again using end which will print it in the same line. Now, let us try to execute this code using some of the sample test cases. Enter a number 9, output is 2, 3, 5, 7 as expected, 15, 2, 3, 5, 7, 11, 13, 3; output is only 2. Now, we also have to check the last two cases which are different as compared to the previous test cases.

Enter number 2, now that is the incorrect output because the problem statement says find all prime numbers less than the entered number and the expected output for this test case is no output, that means we have made a mistake in the Python code. Now, let us check what the mistake is.

(Refer Slide Time: 5:44)


P1.py - C:\IITM\Python\Programs\3.2\P1.py (3.7.4)
File Edit Format Run Options Window Help
IIT Madras
ONLINE DEGREE
Home Tools PowerPoint Present...
Ln: 2 Col: 8

```
num = int(input('Enter a number: '))
if(num >2):
    print(2, end = ' ')
for i in range(3, num):
    flag = False
    for j in range(2, i):
        if(i % j == 0):
            flag = False
            break
        else:
            flag = True
    if(flag):
        print(i, end = ' ')
```

Problem 1: Find all prime numbers less than the entered number

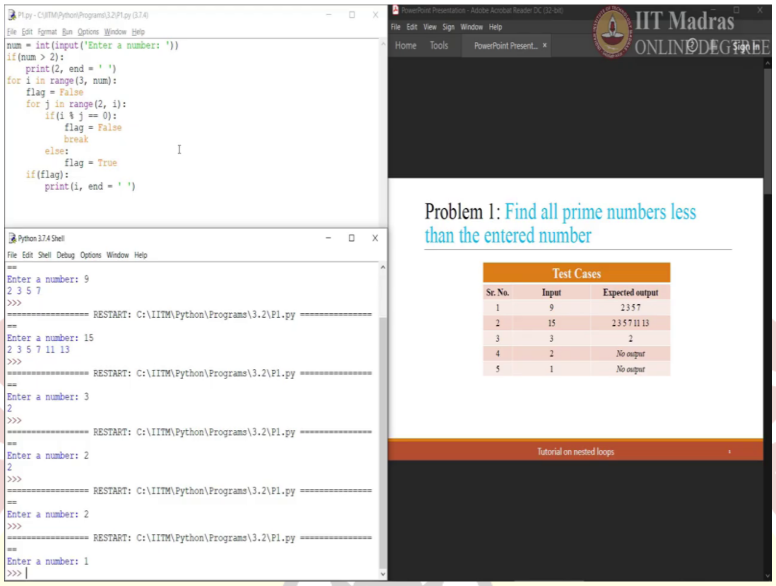
Test Cases

Sr. No.	Input	Expected output
1	9	2 3 5 7
2	15	2 3 5 7 11 13
3	3	2
4	2	*No output*
5	1	*No output*

Tutorial on nested loops

In the first if condition we are checking whether number is greater than equal to 2, as the problem statement says the prime numbers less than the entered number, there we should not use equal to sign over here, it should be number greater than 2. Now, let us try to execute the same code again.

(Refer Slide Time: 6:10)



```
num = int(input('Enter a number: '))
if(num > 2):
    print(2, end = ' ')
for i in range(3, num):
    flag = False
    for j in range(2, i):
        if(i % j == 0):
            flag = False
            break
        else:
            flag = True
    if(flag):
        print(i, end = ' ')
```

```
==
Enter a number: 9
2 3 5 7
>>>
================ RESTART: C:\IITM\Python\Programs\3.2\P1.py ===============
==
Enter a number: 15
2 3 5 7 11 13
>>>
================ RESTART: C:\IITM\Python\Programs\3.2\P1.py ===============
==
Enter a number: 3
2
>>>
================ RESTART: C:\IITM\Python\Programs\3.2\P1.py ===============
==
Enter a number: 2
2
>>>
================ RESTART: C:\IITM\Python\Programs\3.2\P1.py ===============
==
Enter a number: 2
>>>
================ RESTART: C:\IITM\Python\Programs\3.2\P1.py ===============
==
Enter a number: 1
>>> |
```

Problem 1: Find all prime numbers less than the entered number

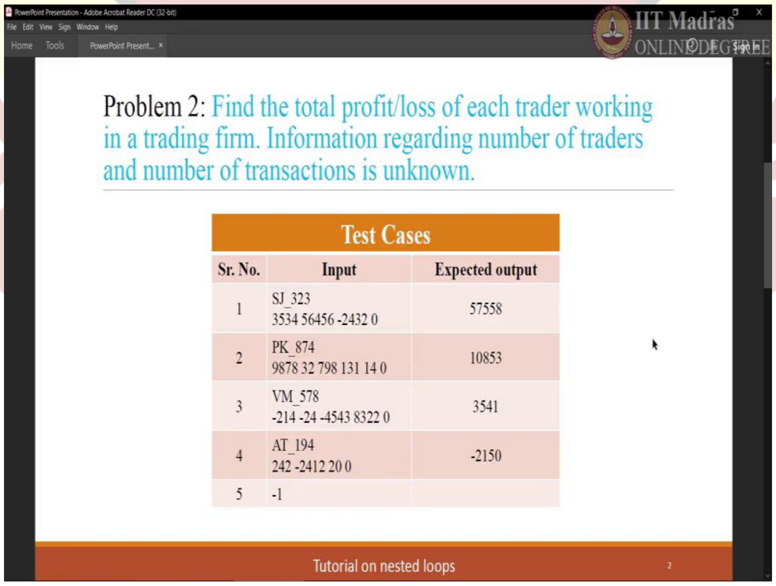
Test Cases

Sr. No.	Input	Expected output
1	9	2 3 5 7
2	15	2 3 5 7 11 13
3	3	2
4	2	*No output*
5	1	*No output*

Tutorial on nested loops

Enter a number 2, no output, which is expected, next enter a number 1, no output once again that was the expected result. The idea behind introducing this particular problem statement was to demonstrate how two for loops can be nested. This was possible because the range function was possible to define in both the cases. In first case the range function was going from 3 to the number whereas in second case it should go from 2 to i whereas i represents the value of the outer iteration. Let us look at a different kind of nesting using while loops.

(Refer Slide Time: 7:07)

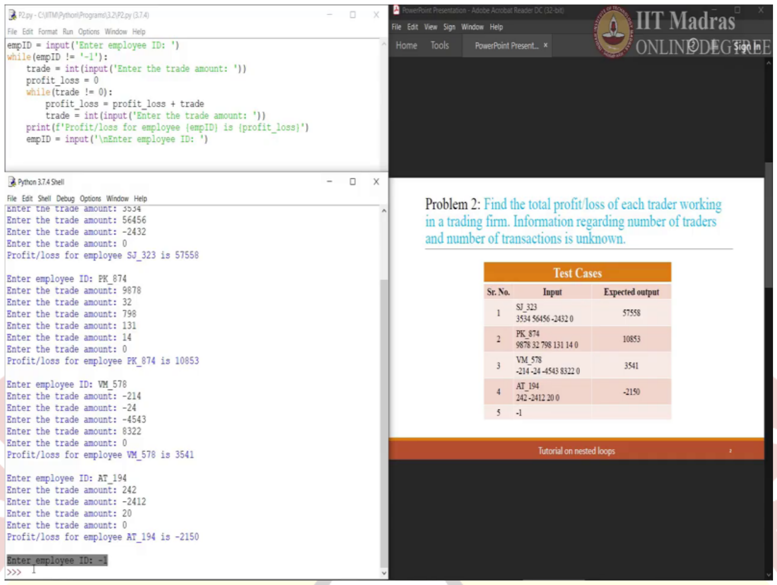


Problem 2: Find the total profit/loss of each trader working in a trading firm. Information regarding number of traders and number of transactions is unknown.

Test Cases

Sr. No.	Input	Expected output
1	SJ_323 3534 56456 -2432 0	57558
2	PK_874 9878 32 798 131 14 0	10853
3	VM_578 -214 -24 -4543 8322 0	3541
4	AT_194 242 -2412 20 0	-2150
5	-1	

Tutorial on nested loops 2



```python
empID = input('Enter employee ID: ')
while(empID != '-1'):
    trade = int(input('Enter the trade amount: '))
    profit_loss = 0
    while(trade != 0):
        profit_loss = profit_loss + trade
        trade = int(input('Enter the trade amount: '))
    print(f'Profit/loss for employee {empID} is {profit_loss}')
    empID = input('\nEnter employee ID: ')
```

```
Enter the trade amount: 3534
Enter the trade amount: 56456
Enter the trade amount: -2432
Enter the trade amount: 0
Profit/loss for employee SJ_323 is 57558

Enter employee ID: PK_874
Enter the trade amount: 9878
Enter the trade amount: 32
Enter the trade amount: 798
Enter the trade amount: 131
Enter the trade amount: 14
Enter the trade amount: 0
Profit/loss for employee PK_874 is 10853

Enter employee ID: VM_578
Enter the trade amount: -214
Enter the trade amount: -24
Enter the trade amount: -4543
Enter the trade amount: 8322
Enter the trade amount: 0
Profit/loss for employee VM_578 is 3541

Enter employee ID: AT_194
Enter the trade amount: 242
Enter the trade amount: -2412
Enter the trade amount: 20
Enter the trade amount: 0
Profit/loss for employee AT_194 is -2150

Enter employee ID: -1
>>>
```

Problem 2: Find the total profit/loss of each trader working in a trading firm. Information regarding number of traders and number of transactions is unknown.

Test Cases

Sr. No.	Input	Expected output
1	SJ_323 3534 56456 -2432 0	57558
2	PK_874 9878 32 798 131 14 0	10853
3	VM_578 -214 -24 -4543 8322 0	3541
4	AT_194 242 -2412 20 0	-2150
5	-1	

Tutorial on nested loops

Find the total profit or loss of each trader working in a trading firm, information regarding number of traders and number of transactions is unknown. Now, if you look at the test cases, first we are getting employee id of the trader, then we are getting different trade values either in terms of profits or loss for that particular trader for whom this particular trade id is.

We are stopping accepting these trade values when we receive 0. As we go on accepting and calculating the total trades for each trader at some point we will receive employee id as minus 1 and that is the place where we should stop the execution of our program. Which means now we require two loops, one loop which will accept the values for all these traders and it will stop when we reach to minus 1 and one more loop which will accept all these trade amounts for each trader until it receives 0.

Therefore, while loop is more suitable solution because we cannot predict how many traders will be there, at the same time we cannot predict how many transactions will be there as given in the problem statement. So, let us try to write a Python program which contains two nested while loops, one for number of traders and second for number of trades.

First we will accept the employee id, then if employee id is not equal to 1 keep accepting the information for new employee every time, then we will accept the trade value, we will declare one variable profit underscore loss as 0, this particular variable will be used to calculate the final amount of trade irrespective of whether it is a profit or a loss.

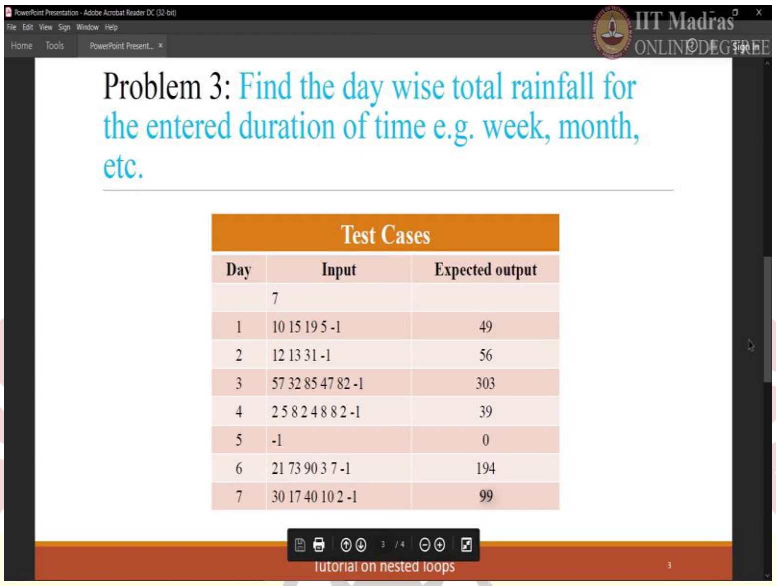
Then, if trade amount is not equal to 0, then we will continue to accept the trade amounts from the trader. Before that we have to add the previous trade value into this particular variable profit underscore loss, this loop will continue until we receive value 0, which is the stopping condition for the trade values.

Once that happens we will print the result of that particular trader using formatted printing like this, where this represents a variable employee id and this represents variable profit loss, at the time of printing the values which are stored against these variables will be replaced and then the statement will be printed.

Once that is executed for one employee then we will accept the employee id of next employee until we get minus 1 as a input. Similarly, for that new employee all these steps will be executed and the final profit or loss value will be printed. Now, let us try to execute this Python code for the given test cases. This will be the output of the given program with respect to given test cases.

First we entered the employee id SJ 323, we got the expected output 57558, then the program continued because we entered PK 874 not minus 1, we kept entering the all trade values and it stopped when we entered 0, we got the output which is 10853, then similar thing happened for this employee VM 578, then for AT 194 and then at the end when we entered the employee id as minus 1, the execution of this Python program stopped, this is exactly what the given problem statement suggests. Now, let us look at a different variation of nested loop.

(Refer Slide Time: 12:20)



Problem 3: Find the day wise total rainfall for the entered duration of time e.g. week, month, etc.

Test Cases

Day	Input	Expected output
	7	
1	10 15 19 5 -1	49
2	12 13 31 -1	56
3	57 32 85 47 82 -1	303
4	2 5 8 2 4 8 8 2 -1	39
5	-1	0
6	21 73 90 3 7 -1	194
7	30 17 40 10 2 -1	99

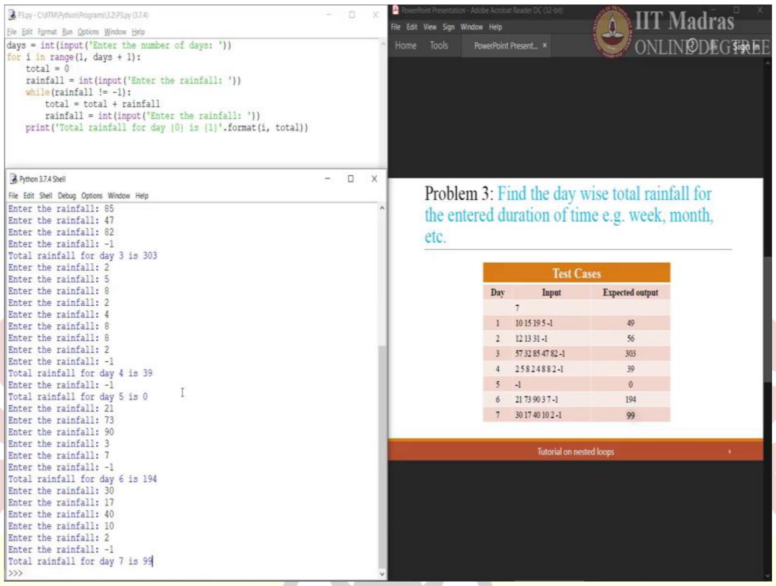
Tutorial on nested loops

Third problem statement find the day wise total rainfall for the entered duration of time. For example, week, month, etcetera. Now, as per the test cases we have to enter number of days and then for each day from 1 to that number n we have to accept different values of rainfall for that particular day. Then we have to keep accepting these numbers till we get minus 1 and the output should be the total rainfall of that particularity.

Therefore, for day 1 the total rainfall is 10 plus 15 plus 19 plus 5 which is 49, we will keep doing this for 7 different days. Hence, we have to accept these values 7 different times, means 7 iterations. As this number is fixed we can clearly decide that for loop is a better suited option for this outer loop, whereas the inner loop should accept these rainfall values till it gets minus 1.

Therefore, while loop is a better suited solution. If you look at day 5 only entered number is minus 1 which means there was no rainfall on that particular day hence, the total is 0. Now, let us try to write a Python program which represents this problem statement using nested while inside a for loop.

(Refer Slide Time: 14:13)



```
days = int(input('Enter the number of days: '))
for i in range(1, days + 1):
    total = 0
    rainfall = int(input('Enter the rainfall: '))
    while(rainfall != -1):
        total = total + rainfall
        rainfall = int(input('Enter the rainfall: '))
    print('Total rainfall for day {0} is {1}'.format(i, total))
```

```
Enter the rainfall: 85
Enter the rainfall: 47
Enter the rainfall: 82
Enter the rainfall: -1
Total rainfall for day 3 is 303
Enter the rainfall: 2
Enter the rainfall: 5
Enter the rainfall: 8
Enter the rainfall: 2
Enter the rainfall: 4
Enter the rainfall: 8
Enter the rainfall: 8
Enter the rainfall: 2
Enter the rainfall: -1
Total rainfall for day 4 is 39
Enter the rainfall: -1
Total rainfall for day 5 is 0
Enter the rainfall: 21
Enter the rainfall: 73
Enter the rainfall: 90
Enter the rainfall: 3
Enter the rainfall: 7
Enter the rainfall: -1
Total rainfall for day 6 is 194
Enter the rainfall: 30
Enter the rainfall: 17
Enter the rainfall: 40
Enter the rainfall: 10
Enter the rainfall: 2
Enter the rainfall: -1
Total rainfall for day 7 is 99
>>>
```

Problem 3: Find the day wise total rainfall for the entered duration of time e.g. week, month, etc.

Test Cases		
Day	Input	Expected output
	7	
1	10 15 19 5 -1	49
2	12 13 31 -1	56
3	57 32 85 47 82 -1	303
4	2 5 8 2 4 8 8 2 -1	39
5	-1	0
6	21 73 90 3 7 -1	194
7	30 17 40 10 2 -1	99

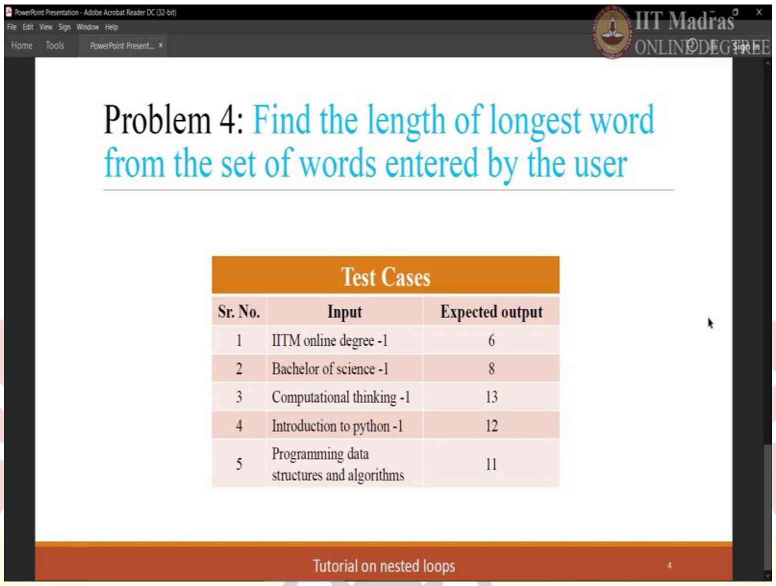
Tutorial on nested loops

We will accept the number of days from the user then we will run the for loop from 1 to days plus 1, 1 to days plus 1 because we are calling first day as day 1 and the last day as day 7 instead of 0 to 6. Declare one variable total and initialize to 0, accept the first value of rainfall for day 1, then keep accepting such rainfall values for each day until we get minus 1, keep adding this rainfall values into variable total whenever we get minus 1 we will stop the inner iteration which is this for loop.

And then this particular print statement will print the total rainfall for day, 0 will be replaced by i and 1 will be replaced by total. Therefore, in case of day 1 it should print total rainfall for day 1 is 49 let us execute this particular Python code with the given test cases.

Enter the number of days 7, then the rainfall for that particular day 10, 15, 19, 5 and to stop it minus 1, hence the total rainfall for day 1 is 49. Then similar for day 2 which is 56, day 3 303, day 4 39, day 5 as we have directly entered minus 1 which means there was no rainfall on that particular day, hence, day 5 0, day 6 194, day 7 99 and that is the place we will stop because the outer for loop was executing from 1 to days plus 1 which are total 7 iterations. Now, let us see the remaining variation of nested loops.

(Refer Slide Time: 16:39)



Problem 4: Find the length of longest word from the set of words entered by the user

Test Cases		
Sr. No.	Input	Expected output
1	IITM online degree -1	6
2	Bachelor of science -1	8
3	Computational thinking -1	13
4	Introduction to python -1	12
5	Programming data structures and algorithms	11

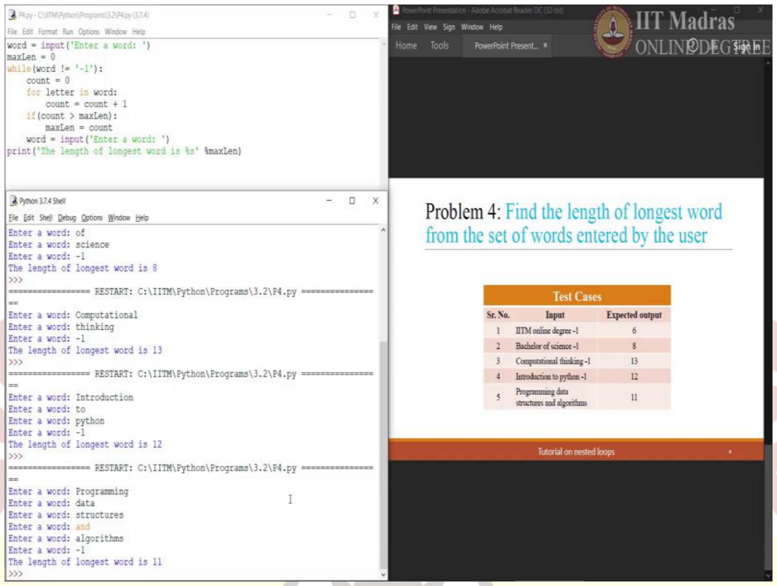
Tutorial on nested loops

Find the length of longest word from the set of words entered by the user. Here we have to enter one word at a time till we get minus 1. Once that happens, then we have to print the length of the longest word which means we have to write one loop which will accept these words one at a time, then the inner loop will calculate the number of letters or number of characters in that particular word which is nothing but the length of the word.

Every time a new word comes we have to compare the length of that new word with the previous maximum value and if the current length is greater than previous max then we will update the max variable with the current value of that word length, which clearly indicates that the outer loop should be while loop because we cannot predict how many words user will enter every time.

For example, for test case 1, 2 and 4 user is entering 3 words, whereas in test case 3 user is entering 2 words and 5 words in test case number 5. Now, let us look at the Python code representing this problem statement.

(Refer Slide Time: 18:23)


P4.py - C:\IITM\Python\Programs\3.2\P4.py (3.7.4)
File Edit Format Run Options Window Help
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
IIT Madras
ONLINE DEGREE

```python
word = input('Enter a word: ')
maxLen = 0
while(word != '-1'):
    count = 0
    for letter in word:
        count = count + 1
    if(count > maxLen):
        maxLen = count
    word = input('Enter a word: ')
print('The length of longest word is %s' %maxLen)
```

```
Enter a word: of
Enter a word: science
Enter a word: -1
The length of longest word is 8
>>>
================ RESTART: C:\IITM\Python\Programs\3.2\P4.py ==============
==
Enter a word: Computational
Enter a word: thinking
Enter a word: -1
The length of longest word is 13
>>>
================ RESTART: C:\IITM\Python\Programs\3.2\P4.py ==============
==
Enter a word: Introduction
Enter a word: to
Enter a word: python
Enter a word: -1
The length of longest word is 12
>>>
================ RESTART: C:\IITM\Python\Programs\3.2\P4.py ==============
==
Enter a word: Programming
Enter a word: data
Enter a word: structures
Enter a word: and
Enter a word: algorithms
Enter a word: -1
The length of longest word is 11
>>>
```

Problem 4: Find the length of longest word from the set of words entered by the user

Test Cases

Sr. No.	Input	Expected output
1	IITM online degree -1	6
2	Bachelor of science -1	8
3	Computational thinking -1	13
4	Introduction to python -1	12
5	Programming data structures and algorithms	11

Tutorial on nested loops

Accept a word from the user, declare one variable called max length initialized to 0 till the entered word is not minus 1 we will continue accepting words then for each word we will count the number of letters or characters in that word which is nothing but the length of the word using a different variation of for which uses the for each feature of for loop which we have seen earlier.

Which means we will iterate over the input word using a letter or a character at a time. Therefore, count should be incremented every time the value of variable later is updated. Once this particular loop ends we will compare the value of count against current max length, if count is greater than max length we will update the max length with the count.

When we get minus 1 as input we will stop the for loop and print final output which says the length of longest word is the value of variable max length. Now, let us execute this particular code using the given test cases.

Enter a word IITM, enter next word which is online, then degree, in order to stop accepting minus 1 then the length of longest word is 6. For second test case, Bachelor of Science, the length of longest word is 8. Then next test case longest is 13, longest is 12 and the last longest is 11.

Based on these four problem statements we can say that all different types of nesting is possible using for as well as while loop, it can be while inside while, for inside for, for inside while or while inside for as per the requirement of the given problem statement we will decide which nesting method is more suitable. Thank you for watching this tutorial, happy learning.