



IIT Madras

ONLINE DEGREE

Programming in Python
Professor Sudarshan Iyengar
Department of Computer Science and Engineering
Indian Institute of Technology, Ropar
Tutorial on While Loop

(Refer Slide Time: 00:15)

Problem 1: Find the factorial of the given number

Test Cases		
Sr. No.	Input	Expected output
1	5	120
2	2	2
3	0	1
4	-7	Not defined

Tutorial on while loop

Hello, Python students. In this tutorial, we will see how while loop is used in Python using some common examples. Problem statement 1, find the factorial of the given number, and some test cases are given. So let us try to convert this particular problem statement into a Python code.

(Refer Slide Time: 00:35)

```
Problem 1.py - C:\Users\Omkar Joshi\Desktop\Problem 1.py (3.7.4)
File Edit Format Run Options Window Help
num = int(input("Enter a number: "))
fact = 1
while (num > 0):
    fact = fact * num
    num = num - 1
print(fact)
```

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:009359112e, 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:/Users/Omkar Joshi/Desktop/Problem 1.py =====
Enter a number: 5
120
>>>
===== RESTART: C:/Users/Omkar Joshi/Desktop/Problem 1.py =====
Enter a number: 2
2
>>>
===== RESTART: C:/Users/Omkar Joshi/Desktop/Problem 1.py =====
Enter a number: 0
1
>>>
===== RESTART: C:/Users/Omkar Joshi/Desktop/Problem 1.py =====
Enter a number: -7
1
>>> |
```

Problem 1: Find the factorial of the given number

Test Cases		
Sr. No.	Input	Expected output
1	5	120
2	2	2
3	0	1
4	-7	Not defined

Tutorial on while loop

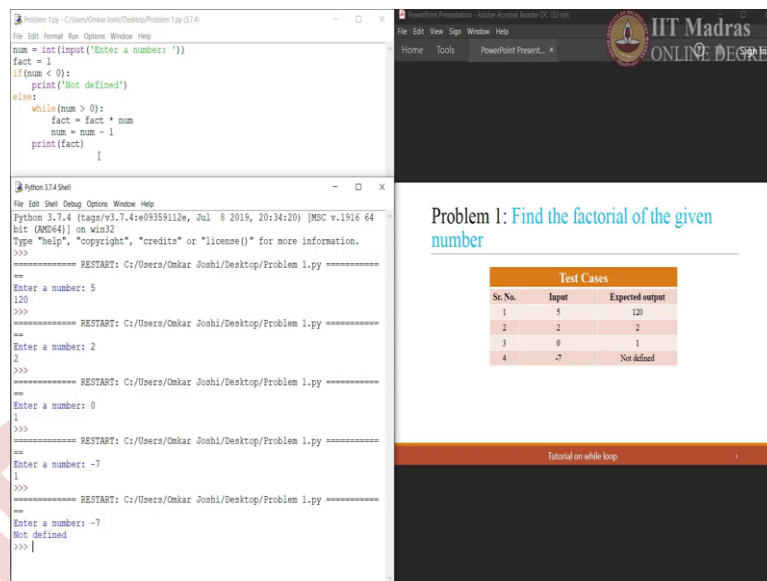
First, we will accept a number from the user using input function. It has to be an integer, so we will convert it to an integer. Next, we will declare a variable called factorial. And let us initialize it to 1 because factorial of 0 is 1. So that is the minimum factorial value we can have. Hence, we will initialize it to 1. As we know factorial of n is computed as n into n minus 1 into n minus 2 up to 1. Hence, we will use that same logic in order to write our while loop.

While num is greater than 0, we will do some calculation. Now what those calculations will be? Factorial is equal to factorial multiplied by number. And the most important step in this particular loop has to be num is equal to num minus 1. This will make sure that we will reduce the number as the iterations go on. Once the entire while loop is finished, then print factorial. Let us try to execute this code using the given test cases.

Enter the number 5, answer is 120. 2 again answer is correct. 0, it is printing 1. Now if you see in this particular case, when we give input as 0, still, it is printing the correct answer, even though the while loop condition says while num is greater than 0, over here, as we are checking the number greater than 0 and 0 is not being greater than 0, while loop will not execute at all. Hence, the factorial variable will be set to 1 and will directly get printed because of this line.

Now let us see what happens with the last test case minus 7, it prints 1. Now over here, the problem is minus 7, it checks whether the minus 7 is greater than 0 or not, which is not the case. Hence, once again the while loop will not execute. And it will directly print the factorial value, which is initially assigned to 1. But as per the given test cases, we are supposed to print factorial of minus 7 as not defined. In order to accommodate these negative numbers, we have to change our code in a such a way so that factorial of all negative numbers will print not defined.

(Refer Slide Time: 04:04)



```
File Edit Format Run Options Window Help
num = int(input("Enter a number: "))
fact = 1
if(num < 0):
    print("Not defined")
else:
    while(num > 0):
        fact = fact * num
        num = num - 1
    print(fact)
I

Python Shell
Python 3.7.4 (tags/v3.7.4:009359112e, 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:/Users/Omkar Joshi/Desktop/Problem 1.py =====
Enter a number: 5
>>>
===== RESTART: C:/Users/Omkar Joshi/Desktop/Problem 1.py =====
Enter a number: 2
>>>
===== RESTART: C:/Users/Omkar Joshi/Desktop/Problem 1.py =====
Enter a number: 0
>>>
===== RESTART: C:/Users/Omkar Joshi/Desktop/Problem 1.py =====
Enter a number: -7
>>>
===== RESTART: C:/Users/Omkar Joshi/Desktop/Problem 1.py =====
Enter a number: -7
Not defined
>>> |
```

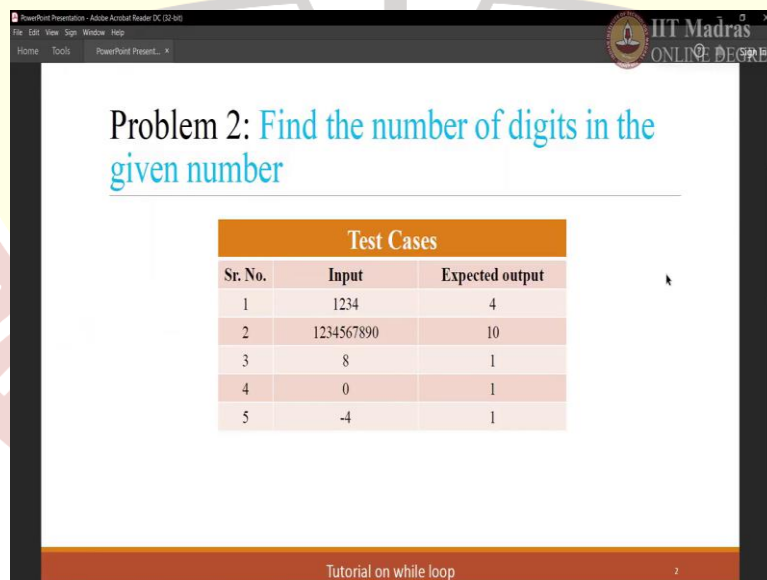
Problem 1: Find the factorial of the given number

Test Cases		
Sr. No.	Input	Expected output
1	5	120
2	2	2
3	0	1
4	-7	Not defined

Tutorial on while loop

For that we have to add 1 more if block. If number is less than 0, which is a negative number, print not defined, else do all these things. Now let us try to execute the updated code with that particular test case, minus 7 not defined. Now the code is correct. Let us go to the next problem statement.

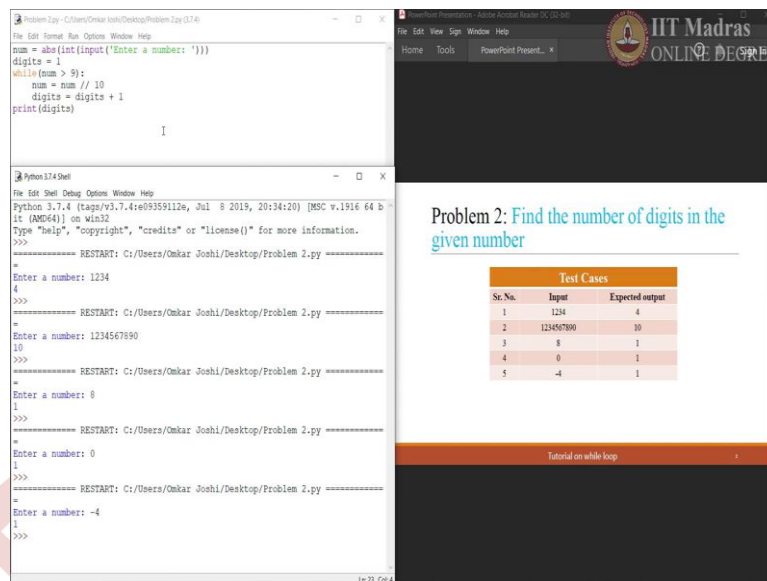
(Refer Slide Time: 04:53)



Problem 2: Find the number of digits in the given number

Test Cases		
Sr. No.	Input	Expected output
1	1234	4
2	1234567890	10
3	8	1
4	0	1
5	-4	1

Tutorial on while loop



Problem statement 2, find the number of digits in the given number. Now let us try to write a Python code for this given problem statement. Once again, we have to accept a number from the user num is equal to int of input of, enter a number. But in this particular problem statement, we have to find number of digits in the given number.

And as you can see, the last test case is looking for number of digits in minus 4. The number of digits are not dependent on the sign of the particular number, it can be a positive number or a negative number, still, the number of digits in this, in the given number are same. Hence, minus 4 or plus 4 both will have only 1 digit, which means we can ignore the sign of the given number.

That can be achieved using a function called absolute, which we can add over here. This particular line has 3 different levels of functions, first, it will take input from the user, then it will convert it into an integer, then it will take the absolute value of that number. And finally, it will get stored in variable num. Let us declare one variable called digits.

Now, the question is what should be the initial value for this particular variable. The lowest possible value for number of digits in any number is 1, because you cannot have a number with 0 digits in it. Hence, we will initialize this particular variable as 1. Next is while loop, when we say digits is equal to 1 that covers all numbers from minus 9 to plus 9, hence, we do not have to do any calculation for those single digit numbers.

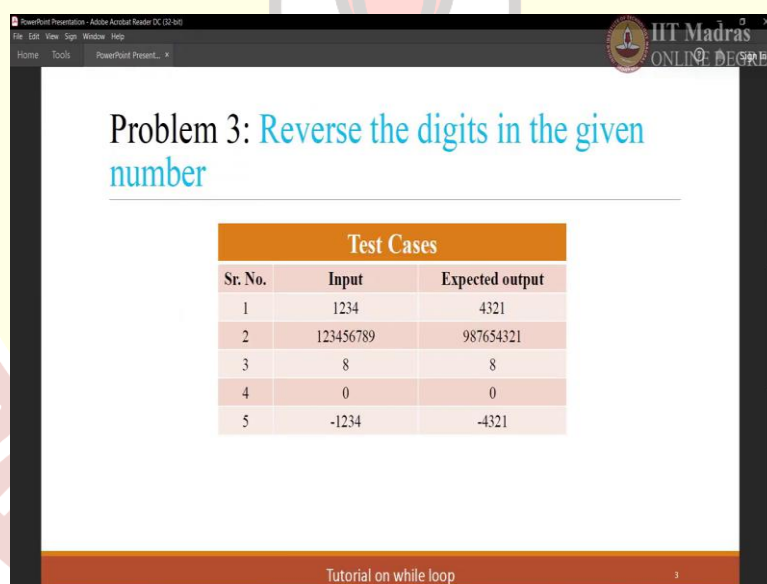
This particular while loop should focus on all those numbers which are greater than equal to 10, means the numbers which are greater than 9. Now, we have to find a way to count the number of digits in the given number. Let us take this example in first test case, it says the

number is 1234. We have to count that each digit in the number separately. So, the question is how to do that?

In order to do something like this, we can use one operator which we have studied earlier, which returns the integer part of the division operation. Hence, this particular operation will reduce down the number from 1234 to 123. If we keep doing this, then it will reduce further to 12 and then 1. For each such iteration, we can increment the variable digits. Once the entire iteration is finished, where number is greater than 9, we will stop this particular loop and print the final result, which is number of digits.

Let us execute this code. Enter a number 1234, answer is 4. Next, we have a very large number. The answer is correct, which is 10. Now let us try some single digit numbers as well. 8, it is printing correct. Let us try 0, that is also correct. Now let us check that interesting part, which is negative numbers minus 4, 1, that is also correct. So, we are getting correct results for negative as well as positive numbers. Let us see next problem statement.

(Refer Slide Time: 09:40)



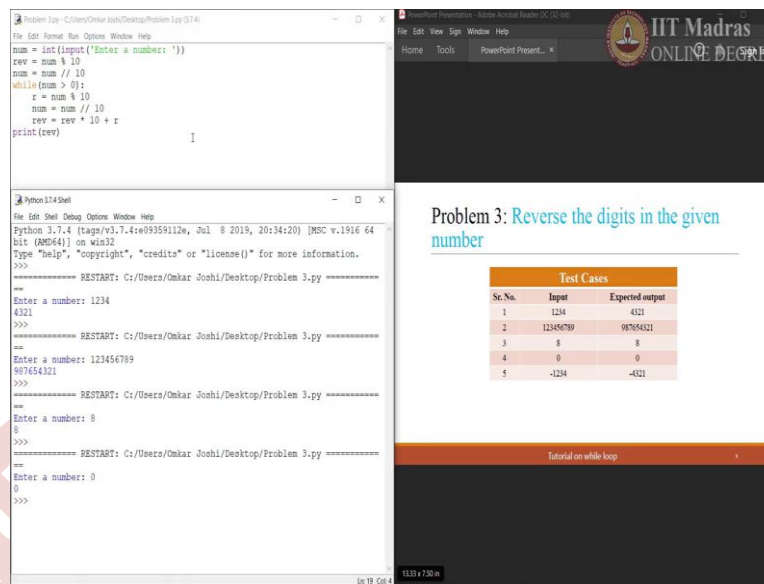
Problem 3: Reverse the digits in the given number

Test Cases		
Sr. No.	Input	Expected output
1	1234	4321
2	123456789	987654321
3	8	8
4	0	0
5	-1234	-4321

Tutorial on while loop

Problem 3, reverse the digits in the given number. This is again a similar problem to the previous one. But here, instead of counting the number of digits, we have to reverse the order of those digits. For example, if the input is 1234 output should be 4321. If you look at the last test case, where a negative number is given, and over there also the expected output is with the reverse digits and the negative sign is also preserved. That is something we have to consider while writing our code. Let us write the Python code to reverse the digits in the given number.

(Refer Slide Time: 10:31)



```
Problem 3.py - C:\Users\Omkar Joshi\Desktop\Problem 3.py (3.7.4)
File Edit Format Run Options Window Help
num = int(input("Enter a number: "))
rev = num // 10
num = num // 10
while (num > 0):
    r = num % 10
    num = num // 10
    rev = rev * 10 + r
print(rev)
```

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:09359112e, 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:/Users/Omkar Joshi/Desktop/Problem 3.py =====
>>>
Enter a number: 1234
4321
>>>
===== RESTART: C:/Users/Omkar Joshi/Desktop/Problem 3.py =====
>>>
Enter a number: 123456789
987654321
>>>
===== RESTART: C:/Users/Omkar Joshi/Desktop/Problem 3.py =====
>>>
Enter a number: 0
0
>>>
===== RESTART: C:/Users/Omkar Joshi/Desktop/Problem 3.py =====
>>>
Enter a number: 0
0
>>>
```

Problem 3: Reverse the digits in the given number

Test Cases		
Sr. No.	Input	Expected output
1	1234	4321
2	123456789	987654321
3	0	0
4	0	0
5	-1234	-4321

Tutorial on while loop

num is equal to int of input of enter a number. Let us define one variable which will store a number in the reverse order. Now, the question is what should be the initial value of this particular variable reverse. If you remember from the previous problem statement, we use integer division operator to find the integer quotient part of the division operation we can do similar thing over here, but over here that remainder part is also important, because in case of 1234, we will require that 4 digit to come in this reverse variable at the first position, then 3, 2 and 1.

How to extract that number 4 from 1234 is the question? This can be done using the modulo operator, which we have seen earlier, $\text{num} \bmod 10$ this particular operation will give output which is 4 and it will be stored in variable reverse. After this, that number 4 is not required, and we can reduce down the number to 123. We have to repeat the same operation until all the digits in the given numbers are reversed.

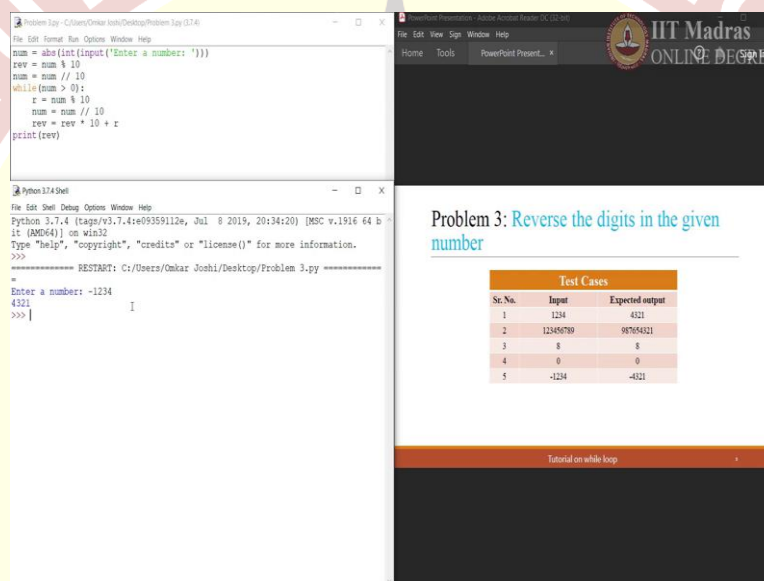
In order to do so, we will write a loop, while num is greater than 0. If we use this reverse is equal to $\text{num} \bmod 10$ statement again over here, that will override the current value of a reverse variable which we do not want. Somehow, we have to retain the current value 4, in addition to that, we have to add the next digit, which is 3. To get that 43, first we require 3 which can be extracted using r is equal to $\text{num} \bmod 10$.

Now this r variable will store the remainder of this particular operation, then num is equal to num divided by 10, then we will again reduce down the number from 123 to 12. Here this particular reverse variable holds value 4, whereas the variable r stores value 3. Somehow, we have to merge these two together and make it 43 into this reverse variable. It can be achieved

using reverse is equal to reverse multiplied by 10 plus r. This same step will be useful in the next iteration as well, where we will have a reverse value as 43 and r will hold 2.

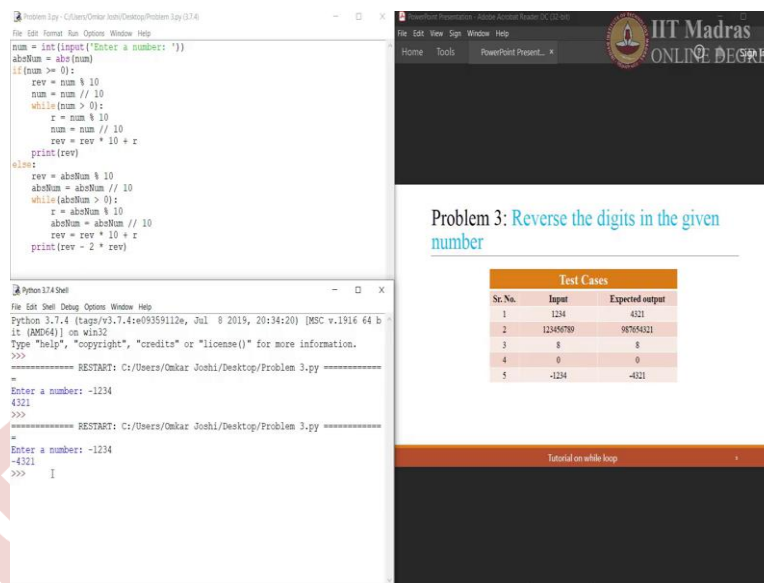
Hence, we can add 43 into 10 plus 2, it will give us 432 and in the last step it will give us 4321 which is the expected output. Hence, we can print this particular answer. Now let us test it with all positive inputs. 1234 answer we are getting the expected one. Now let us try it with that big number. Once again, we are getting the correct answer. Let us try it with single digits also, 8, 0. As the code is working fine for all positive numbers, we have to modify it in such a way so that it can work for negative numbers as well.

(Refer Slide Time: 15:29)



For negative numbers, we can take the absolute value of the user input as we have done in the previous program. After this, let us try to execute what happens with minus 1234. It prints the number in the reverse order, but negative sign is omitted. This is happening because we have taken the absolute value of the input. Hence, whenever we execute the remaining part of the code, the program is not aware whether the original number was 1234 or minus 1234. Hence taking this absolute value at the initial stage is a bad idea, we have to preserve the original input as well.

(Refer Slide Time: 16:23)



```
num = int(input("Enter a number: "))
absNum = abs(num)
if (num >= 0):
    rev = num % 10
    num = num // 10
    while (num > 0):
        r = num % 10
        num = num // 10
        rev = rev * 10 + r
    print(rev)
else:
    rev = absNum % 10
    absNum = absNum // 10
    while (absNum > 0):
        r = absNum % 10
        absNum = absNum // 10
        rev = rev * 10 + r
    print(rev - 2 * rev)
```

Python 3.7.4 Shell

```
Python 3.7.4 (tags/v3.7.4:re09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 b
it (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/Omkar Joshi/Desktop/Problem 3.py =====
>>>
Enter a number: -1234
4321
>>>
===== RESTART: C:/Users/Omkar Joshi/Desktop/Problem 3.py =====
>>>
Enter a number: -1234
-4321
>>>
I
```

Problem 3: Reverse the digits in the given number

Test Cases		
Sr.No.	Input	Expected output
1	1234	4321
2	123456789	987654321
3	8	8
4	0	0
5	-1234	-4321

Tutorial on while loop

In order to do that, we can declare one variable called absolute number. Then we will take the absolute value of the input number and remove this absolute from the first line. This will make sure that we have two different variables storing two different values. This number variable will store the actual user input, whether it is a positive or a negative and then absolute number is another variable, which will store the absolute value of the given input.

As we have seen earlier, this particular code executes correctly for all positive numbers. And that can be checked using a if condition which says if number is greater than equal to 0, execute this particular code block. But now, in case of a negative number, we have to opt for a different strategy which we will use in a else block.

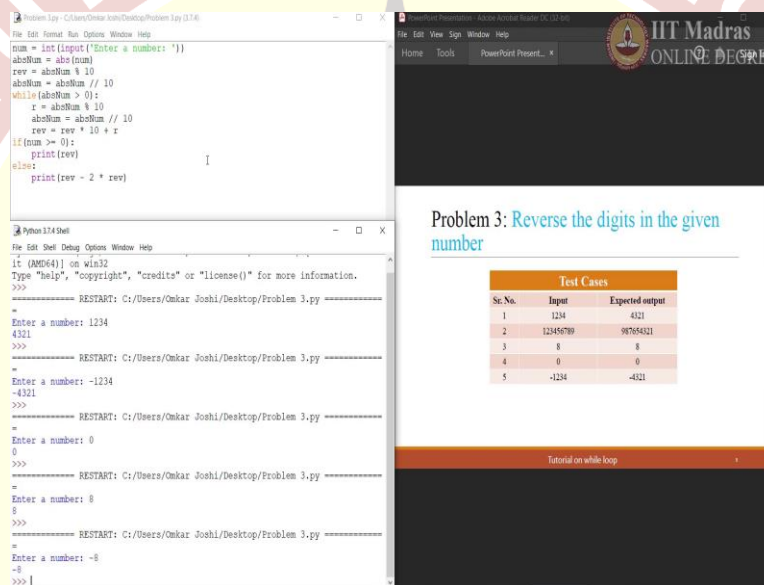
Now in else block, for negative numbers instead of the original number, we will focus on absolute number. Hence reverse is equal to absolute number mod 10, absolute number is equal to absolute number integer division 10, while absolute number greater than 0, execute r is equal to absolute number mod 10, absolute number is equal to absolute number integer division 10. And finally, the reverse is equal to reverse multiplied by 10 plus remainder r. And then the final print statement is the place where we have to add the negative sign.

Now, how to convert the 4321 into minus 4321? That is the question. In order to do this, we can do something like this, reverse minus 2 multiplied by reverse. This will subtract the same number from itself twice, that will give us the same number on the negative side, let us try to execute this particular code, minus 1234. Now, we are getting the correct answer.

But if you look at this particular code, calculation logic used in if as well as else block are almost identical for these particular lines. Here we use num variable and here we have used absolute num variable, apart from that the entire code is same. Hence, this particular code can be taken out of this if else and can be executed irrespective of whether a number is greater than equal to 0.

Because in case of positive input the number as well as absolute number will have same value. Hence, this particular code block can be executed for positive numbers as well. Hence, we will take this part out of the else block and will execute as a common content.

(Refer Slide Time: 21:22)



```

num = int(input("Enter a number: "))
absNum = abs(num)
rev = absNum % 10
absNum = absNum // 10
while(absNum > 0):
    r = absNum % 10
    absNum = absNum // 10
    rev = rev * 10 + r
if(num >= 0):
    print(rev)
else:
    print(rev - 2 * rev)

```

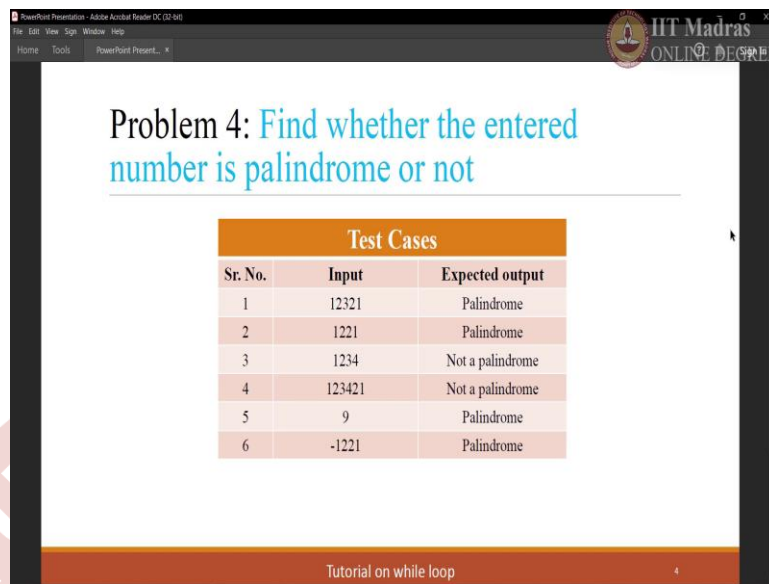
Problem 3: Reverse the digits in the given number

Sr. No.	Input	Expected output
1	1234	4321
2	123456789	987654321
3	8	8
4	0	0
5	-1234	-4321

As we have that as a common content, we do not require to execute these particular steps in if as well as else. Now, the calculation will happen irrespective of whether number is positive or negative, then while printing the answer, we will check whether the number is positive or a negative. And accordingly, we will print either print reverse, or print reverse minus 2 into reverse. Let us try this code with all the given test cases, 1234, that is giving correct answer.

Let us try a negative number also minus 1234, that also giving correct answer. Let us try some single digit numbers also, 0 that is correct. 8 is also correct, minus 8 is also correct. Now, this particular program is giving us the correct outputs for all the given test cases. And one thing we should remember over here is if there is some code block like this one, which can be taken out of if else and executed commonly, then we should opt for that because it optimizes the given code. Now, let us move to next problem statement.

(Refer Slide Time: 23:12)



Problem 4: Find whether the entered number is palindrome or not

Test Cases		
Sr. No.	Input	Expected output
1	12321	Palindrome
2	1221	Palindrome
3	1234	Not a palindrome
4	123421	Not a palindrome
5	9	Palindrome
6	-1221	Palindrome

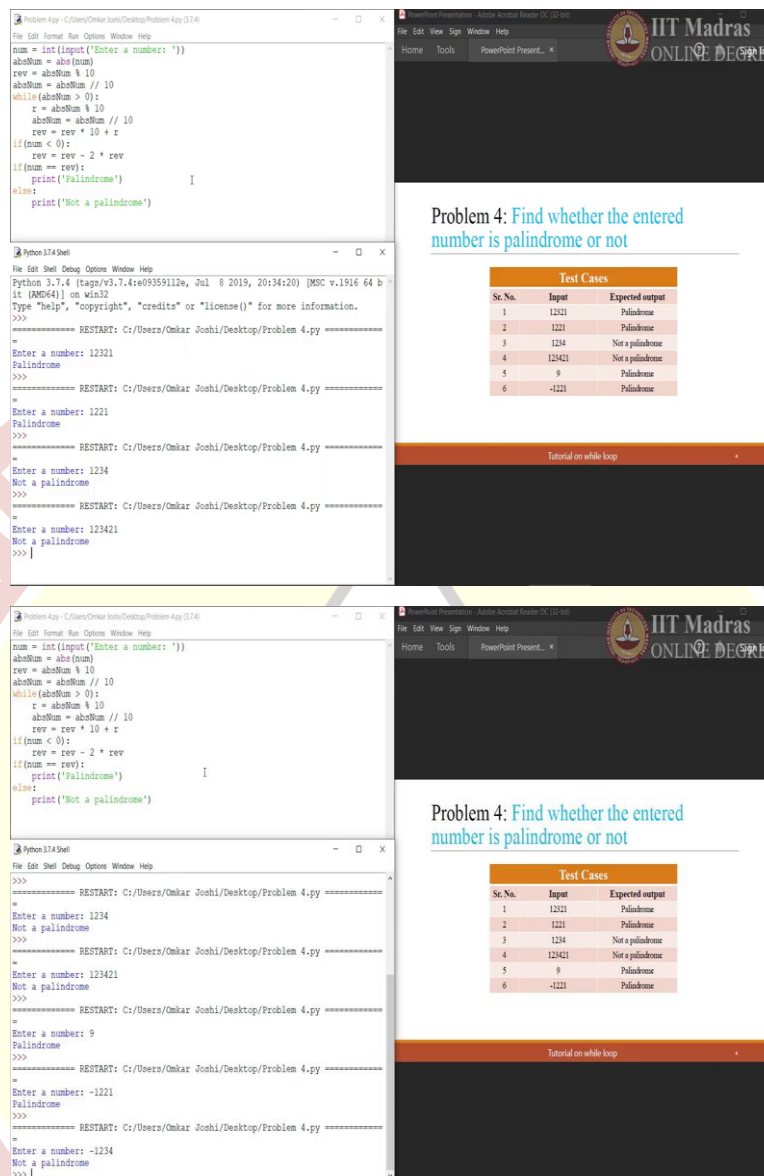
Tutorial on while loop 4

Find whether the entered number is palindrome or not, palindrome is a special type of a number which is exactly equal to its original number, even after reversing the digits. For example, the first test case 12321. If we reverse this particular number, still we will get the same number as the output. Hence, this particular number is considered as palindrome.

Similarly, the second test case 1221 or the last test case, where the input number is a negative number, but if we reverse the digits of the number, still we will get the output as minus 1221. Also, if you look at the fifth test case, it has a single digit and still it says the expected output is palindrome, which means a numbers with single digits are by default palindromes.

Based on this information, we can say that in order to find whether the number is palindrome or not, first we have to find the reverse number of the given input, which we have done in the previous question. Now, we can use that same code and modify it in such a way to print whether the entered number is palindrome or not. Let us try that.

(Refer Slide Time: 24:54)



The top-left screenshot shows a Python script in a text editor. The script takes an input number, reverses its digits, and checks if it is a palindrome. The top-right screenshot shows a presentation slide titled "Problem 4: Find whether the entered number is palindrome or not" with a table of test cases. The bottom-left screenshot shows the Python shell output for the script, demonstrating the reversal of numbers 12321, 1221, 1234, 123421, and -1221. The bottom-right screenshot is another view of the presentation slide.

Problem 4: Find whether the entered number is palindrome or not

Sl. No.	Input	Expected output
1	12321	Palindrome
2	1221	Palindrome
3	1234	Not a palindrome
4	123421	Not a palindrome
5	9	Palindrome
6	-1221	Palindrome

Tutorial on while loop

This is the Python code which we wrote for previous problem statement where we reversed the digits in the entered number. Now, we have to modify this particular code in such a way where we will check whether the entered number is a palindrome or not. To find whether the number is palindrome or not, we require two different numbers, the original input and the number in the reverse order.

But in case of a negative number, reverse variable will hold a number with reverse digits, but it will not have the negative sign. So, we will add one if statement, which will take care of this particular problem, if number is less than 0, then reverse is equal to reverse minus 2 multiplied by reverse. The same logic what we applied earlier; we will use that in order to update the value of a reverse variable if the input number is a negative one.

After this, we simply have to compare the number with its reversed value, which can be done over here as number equal to reverse, if that is the case, then we will consider it as palindrome; if not, then we consider it as not a palindrome. First test case 12321 It is a palindrome. 1221 that is also palindrome. 1234, that is not a palindrome. 123421 that is also not a palindrome.

Then a single digit, which is always palindrome. Then try some negative numbers also 1 minus 1221, that is a palindrome; minus 1234 not a palindrome. This covers all the test cases and all different possibilities of inputs. Hence, we can say that the current program find whether the input number is palindrome or not, is correct. Thank you for watching this tutorial. Happy learning!

