

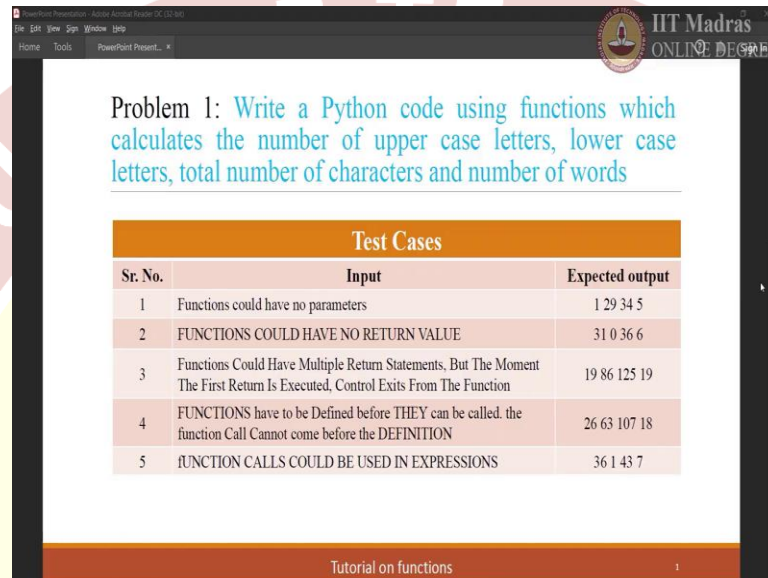


# IIT Madras

ONLINE DEGREE

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

(Refer Slide Time: 00:16)



Problem 1: Write a Python code using functions which calculates the number of upper case letters, lower case letters, total number of characters and number of words

Test Cases		
Sr. No.	Input	Expected output
1	Functions could have no parameters	1 29 34 5
2	FUNCTIONS COULD HAVE NO RETURN VALUE	31 0 36 6
3	Functions Could Have Multiple Return Statements, But The Moment The First Return Is Executed, Control Exits From The Function	19 86 125 19
4	FUNCTIONS have to be Defined before THEY can be called. the function Call Cannot come before the DEFINITION	26 63 107 18
5	FUNCTION CALLS COULD BE USED IN EXPRESSIONS	36 1 43 7

Tutorial on functions 1

Hello Python students. In this tutorial, we will see functions or in fact more precisely user defined functions. First problem statement, write a Python code using functions which calculates the number of uppercase letters, lowercase letters, total number of characters and number of words. Here, few test cases are given. Each given test case specifies a statement regarding functions. For example, test case one functions could have no parameters.

In this case, the expected output is 1, 29, 34, 5 in the same order as mentioned in the problem statement, 1 uppercase letter, 29 lowercase letters, 34 number of characters, and 5 total number of words. Now, looking at these numbers, one may wonder why number of characters are higher than the sum of uppercase letters and lowercase letters. This is because when we count number of characters in a sentence or in a string, we count special characters, including space between two different words. Now, let us write a Python program with functions which will calculate all these different characteristics of the given input.

(Refer Slide Time: 1:54)

```
def upper(s):
    upper = 0
    for c in s:
        if(c.isupper()):
            upper += 1
    return(upper)

def lower(s):
    lower = 0
    for c in s:
        if(c.islower()):
            lower += 1
    return(lower)

def characters(s):
    chars = 0
    for c in s:
        chars += 1
    return(chars)

def words(s):
    words = 1
    for c in s:
        if(c == ' '):
            words += 1
    return(words)

sentence = input("Enter the sentence: ")
uletters = upper(sentence)
print(f"Total number of upper case characters: {uletters}")
lletters = lower(sentence)
print(f"Total number of lower case characters: {lletters}")
chars = characters(sentence)
print(f"Total number of characters: {chars}")
words = words(sentence)
print(f"Total number of words: {words}")
```

Problem 1: Write a Python code using functions which calculates the number of upper case letters, lower case letters, total number of characters and number of words

Test Cases		
Sl. No.	Input	Expected output
1	FUNCTIONS COULD HAVE NO RETURN VALUE	1 29 34 5
2	FUNCTIONS COULD HAVE MULTIPLE RETURN STATEMENTS, BUT THE MOST THE FIRST RETURN IS ENCLOSED, CONTROL EXITS FROM THE FUNCTION	31 0 36 6
3	FUNCTIONS HAVE TO BE DEFINED BEFORE THEY CAN BE CALLED, THE FUNCTION CALL CANNOT COME BEFORE THE DEFINITION	19 60 125 19
4	FUNCTIONS HAVE TO BE DEFINED BEFORE THEY CAN BE CALLED, THE FUNCTION CALL CANNOT COME BEFORE THE DEFINITION	26 63 107 18
5	FUNCTION CALLS COULD BE USED IN EXPRESSIONS	36 1 43 7

Tutorial on functions

We will accept an input from the user and store it in a variable called sentence. This sentence variable will store the entire input as a single string. For example, in case of first test case, the sentence variable will store a string and the value for that string will be functions could have no parameters.

Now, next part is to write a function to calculate total number of uppercase letters in the given input. Let us write one function called upper. To this particular function, we will pass this particular variable sentence and this function is supposed to return back the total number of uppercase letters in the sentence which can be stored in a variable uLetters as an uppercase letters. Once we have the total count for uppercase letters, we can print it using formatted printing. But the most important part with respect to this particular code is to write this upper function.

Now, let us see how this function can be written. As we have studied earlier, in Python, a function is written using keyword def, followed by function name, followed by list of parameters. In this particular case, we will call a function upper, will pass this particular parameter sentence, whatever value which is stored in this variable sentence will be passed to this particular variable s. And in this particular function upper, we can use this variable s to do our computation.

Now, let us look at the logic written inside this particular function upper. We are declaring one variable called upper, initializing it to 0 which makes it integer variable. This particular variable will be used to store the total number of uppercase characters in this sentence then we will use this for each feature of for loop which will iterate over the string character by character. Therefore, this variable `c` will hold one character at a time from string variable `s`. Every time value in variable `c` updates we will check whether `c` is upper or not.

Over here, `isupper` is a library function which we have studied earlier. This particular function will return true if value in variable `c` is an uppercase letter. Otherwise, it will return false. Whenever this particular function call returns true, we will increment the value of variable upper. Now, this particular operator might look odd, but it is referred as shorthand operator, which means upper is equal to upper plus 1. This is easier way to increment the value of any variable, which means we will increment the upper variable every time when `c dot isupper` returns true.

At the end, we will return the variable upper when for loop terminates. This particular return statement will return one integer value back to the place from where this function was called, which is over here, which means now whatever value was there in this particular variable upper is now stored in this particular variable `uLetters`, which then can be used to print the uppercase letters along with the message which says total number of uppercase characters.

Next part of this particular problem statement is to find lowercase letters in the given input. Just like the code which we return to find uppercase letters, we have to write similar code to find lowercase letters in the given input. So, the first step we will declare one variable `lLetters` as in lowercase letters and initialize it with a function call, which says lower and will pass this particular input sentence as a parameter to that particular function. Once we get the value of lowercase letters in the given input, we will print that particular number along with the message which says total number of lowercase characters.

As we have seen earlier, the most important part with respect to this particular code is to write this particular lower function. Let us try to write that particular function. Just like upper function, we will define the function `def lower` and `s` as a parameter which will store the input which is coming from sentence then we will iterate over that sentence character by character using for

each functionality, then in each block we will call another library function `islower`. This particular function is exactly like what we did earlier using `isupper`.

Only difference is this particular function returns true if the character is a lowercase letter. If that is the case, then we will increment the variable `lower`. Once the for loop terminates, we will return the variable `lower` which holds the total number of lowercase letters in that particular sentence, which will return back value to this variable `lLetters` which can be printed.

Now, as per the problem statement, we have to count total number of letters in the given sentence. Let us try to do that as our next function. Total number of characters is equal to character function with a parameter sentence, which means now we have to write one function called `characters` which will count the total number of characters in the sentence and it should return back total number of characters which can be then stored in this particular variable `chars` and then we can print it along with the message total number of characters and the value once again using the formatted printing.

Let us write that particular function `characters`. If you compare this particular `characters` function with `lower` or `upper`, you will see it is very similar, except there is only one change, which is this particular if block is missing. Because now we have to increment this variable `chars` every time for every character in the sentence irrespective of whether it is a lowercase character, uppercase character, a space or even a special character like comma, full stop, etcetera. Once again, when this particular for loop terminates, we will return the count which is stored in variable `chars` back to this variable `chars`, which will get printed.

Now, let us move to last part of this particular problem statement where we are supposed to count total number of words in the given input. Once again, we will declare one variable `words` and against that, we will call one function called `words`, which will accept this input sentence as a parameter and then we will print the total number of words using formatted printing. Let us try to write that particular `words` function.

Over here, we are declaring one variable `words` and initializing it with 1 instead of 0, because any sentence will have at least one word. Then once again, we will use this for each functionality of for loop, then we will use this particular if condition which is very interesting, because it is checking the value of character against these empty quotes. But if you observe carefully, there is

a space between these empty quotes, which means we are actually checking whether the value stored in this particular character is space or not.

If the value which is stored in this character is space, which means the if condition is true, then we will increment variable words, which means whenever we encountered a space, we will increment words, because space indicates that the previous word finished and a new word is going to start. Once again, we will return this particular variable words which will return back to the function call and the value of words variable over here will get updated, which can be used to print the actual value using formatted printing.

(Refer Slide Time: 12:43)

Sl. No.	Input	Expected output
1	Functions could have no parameters	1 29 34 5
2	FUNCTIONS COULD HAVE NO RETURN VALUE	31 9 36 6
3	Functions Could Have Multiple Return Statements, But The Moment The First Return Is Executed, Control Exits From The Function	19 86 125 19
4	FUNCTIONS have to be Defined before THEY can be called. the function Call Cannot come before the DEFINITION	26 63 187 18
5	FUNCTION CALLS COULD BE USED IN EXPRESSIONS	36 1 43 7



The screenshot shows a Python IDE on the left and a presentation slide on the right. The Python IDE displays the execution of a program that counts the number of upper case characters, lower case characters, total number of characters, and total number of words in a sentence. The program is executed multiple times with different input sentences, and the output is displayed in the console. The presentation slide on the right shows a problem statement and a table of test cases.

**Problem 1: Write a Python code using functions which calculates the number of upper case letters, lower case letters, total number of characters and number of words**

Sl. No.	Input	Expected output
1	Functions could have no parameters	1 29 34 5
2	FUNCTIONS COULD HAVE NO RETURN VALUE	31 0 36 6
3	Functions Could Have Multiple Return Statements, But The Moment The First Return Is Executed, Control Exits From The Function	19 86 125 19
4	FUNCTIONS have to be Defined before THEY can be called, the function Call Cannot come before the DEFINITION	26 63 107 18
5	FUNCTION CALLS COULD BE USED IN EXPRESSIONS	36 1 43 7

Now, let us try to execute this code using the given test cases. Let us execute the first test case. Functions could have no parameters, total number of uppercase characters 1, lowercase characters 29, number of characters 34, number of words 5. Second test case, functions could have no return value 31, 0, 36, 6. Third test case, functions could have multiple return statements, but the moment the first return is executed control exists from the function 19, 86, 125 and 19.

Fourth test case, functions have to be defined before they can be called. The function call cannot come before the definition 26, 63, 107 and 18. Then the last test case, function calls could be used in expressions 36, 1, 43, 7. As we are getting all the expected outputs for the given test cases, let us move on to next problem statement.

(Refer Slide Time: 14:07)

Problem 2: Write a Python code using functions to calculate area and perimeter of circle and rectangle

Test Cases					
Sr. No.	Input	Expected output	Sr. No.	Input	Expected output
1	"circle" "area" 7	154 sq. units	2	"circle" "perimeter" 7	44 units
3	"rectangle" "area" 15 10	150 sq. units	4	"rectangle" "perimeter" 15 10	50 units
5	"exit"	Stop execution			

Tutorial on functions

Second problem statement, write a Python code using functions to calculate area and perimeter of circle and rectangle. In this particular statement, we have to calculate two different properties perimeter and area of two different polygons, circle and rectangle. If we look at the test cases, the program expect us to take polygon name and the property of the polygon as input along with either a radius or a dimension of the rectangle. Depending on these inputs, we are supposed to calculate the respective property and print the result.

For example, if input is circle, area and 7, then the expected output is 154 square units, because a circle with radius 7 units will have area as 154 square units. Similarly, if we looked at the second test case, the input is circle, perimeter and value 7. The expected output is 44 units. In this case, we are supposed to calculate perimeter, because it is mentioned in the input. Similarly, in test case 3 and 4, the first level of input is rectangle. And then the second level of input is the property of that rectangle, either its area or a perimeter. Then, based on the values for length and breadth, we will either calculate area which is 150 square units or perimeter which is 50 units.

The most interesting test case over here is the fifth one, which says if the input is exit, then the expected output is to stop the execution, which means whenever we enter input as exit, it should stop the execution of our Python program. Based on these test case cases, we can figure out that this is a different type of program as compared to what we have seen earlier. And this particular type of program is called a menu driven program, where a standardized menu will be given to us



and we have to choose a particular option from the given menu. And based on that option, a specific function will be executed from the Python code.

Now, let us try to write the Python code for this particular problem statement. First, we will start with a standard program without any menu, then we will try to convert the standard program into a menu driven code.

(Refer Slide Time: 17:22)

**Problem 2: Write a Python code using functions to calculate area and perimeter of circle and rectangle**

Test Cases					
Sr. No.	Input	Expected output	Sr. No.	Input	Expected output
1	"circle" "area"	154 sq. units	2	"circle" "perimeter"	44 units
3	"rectangle" "area"	150 sq. units	4	"rectangle" "perimeter"	50 units
5	"exit"	Stop execution			

```

***Problem 2: Write a Python code using functions to calculate area and perimeter of circle and rectangle***
***Approach 1: Standard code***
PI = 22 / 7
def circle_area(r):
    return (PI * r * r)
def circle_perimeter(r):
    return (2 * PI * r)
def rectangle_area(l, b):
    return (l * b)
def rectangle_perimeter(l, b):
    return (2 * (l + b))
r = float(input("Enter the radius of the circle: "))
cArea = circle_area(r)
print(f"Area of circle with radius (r) = {cArea} sq. units")
cPerimeter = circle_perimeter(r)
print(f"Perimeter of circle with radius (r) = {cPerimeter} units")
l = float(input("Enter the length of the rectangle: "))
b = float(input("Enter the breadth of the rectangle: "))
rArea = rectangle_area(l, b)
print(f"Area of rectangle with length (l) and breadth (b) = {rArea} sq. units")
rPerimeter = rectangle_perimeter(l, b)
print(f"Perimeter of rectangle with length (l) and breadth (b) = {rPerimeter} units")
    
```

```

***Problem 2: Write a Python code using functions to calculate area and perimeter of circle and rectangle***
***Approach 2: Menu driven code***
PI = 22 / 7
def circle_area(r):
    return (PI * r * r)
def circle_perimeter(r):
    return (2 * PI * r)
def rectangle_area(l, b):
    return (l * b)
def rectangle_perimeter(l, b):
    return (2 * (l + b))
polygon = ""
while (polygon != "exit"):
    print("\nPOLYGONS (circle/rectangle/exit)")
    polygon = input("Choose the polygon type or exit: ")
    if (polygon == "circle"):
        r = float(input("Enter the radius of the circle: "))
        while (property == ""):
            print("\nCIRCLE PROPERTIES (area/perimeter/back)")
            property = input("Choose the circle property or go back: ")
            if (property == "area"):
                cArea = circle_area(r)
                print(f"Area of circle with radius (r) = {cArea} sq. units")
            elif (property == "perimeter"):
                cPerimeter = circle_perimeter(r)
                print(f"Perimeter of circle with radius (r) = {cPerimeter} units")
            elif (property == "back"):
                break
        else:
            print("Please select the correct polygon property")
            property = ""
    elif (polygon == "rectangle"):
        l = float(input("Enter the length of the rectangle: "))
        b = float(input("Enter the breadth of the rectangle: "))
        rArea = rectangle_area(l, b)
        print(f"Area of rectangle with length (l) and breadth (b) = {rArea} sq. units")
        rPerimeter = rectangle_perimeter(l, b)
        print(f"Perimeter of rectangle with length (l) and breadth (b) = {rPerimeter} units")
    else:
        print("Please select the correct polygon type or exit")
    polygon = input("Choose the polygon type or exit: ")
    
```

Now, let us look at this particular Python code which is a standard approach or approach one where we will not have any menu. First, we have declared one variable pi, and we have assigned

value as 22 by 7. Then we have written four different functions, first to calculate area of a circle as  $\pi$  into  $r$  into  $r$ ,  $r$  being the radius of the circle; second, to calculate the perimeter of the circle  $2$  into  $\pi$  into  $r$ ; next one, area for rectangle which is length into breadth, next perimeter of the rectangle, which is  $2$  into length plus breadth.

In order to call these four different functions, we have accepted the value for radius from user. Then using that particular value  $r$ , we call this function `circle underscore area`, whatever value it returns that get, that will get stored in variable `c area` and we will print it using this particular formatted print statement. Similarly, we will call `circle underscore perimeter` function which will calculate the perimeter of the circle with the given value for radius. Calculated result will be return to variable `cPerimeter` and then we can print it using this particular formatted print.

For rectangle, we will accept two different values from user, first being length and second is breadth. Once both these values are available with us, we will call this particular function `rectangle underscore area` with those two values  $l$  and  $b$  as parameters. The computed result will be return back to this variable `r area`, which we will print over here. Next, we will call `rectangle underscore perimeter` function with the same values  $l$  comma  $b$  which will give us the perimeter of that particular rectangle in variable `r perimeter`, which you will print last sentence.

As you can see, this is a standard code where no menu is being displayed, which will ask a user to enter these kinds of inputs, which says circle or area or rectangle or perimeter and so on. In order to achieve this particular problem statement, we have to make some modifications into this particular standard code and include this particular feature called menu driven program. Now, let us try to write a Python code for that particular program.

In this approach two menu driven code, we will retain these four functions as it is. We will try to write a printed statements in such a way which will print the menu and also indicate the user to enter the right choice from the given list of options. In this case, the first level of option should be whether the polygon is circle or a rectangle or do we want to exit the code. Let us create a structure where we will ask the user to enter the name of the polygon or the word exit if user wish to stop the execution of the code.

In order to achieve that, first, we will declare one string variable called `polygon` and we will initialize it with a empty string. Then as we discussed earlier, this particular program should

continue its execution till user enters a specific keyword which is exist. Therefore, we will write our code inside a while loop which will continue to execute till the value of variable polygon is not equal to exit.

After this, as discussed, first, we have to print a menu which will have a list of options from which a user can choose a specific polygon or exit. This print statement will print those different options, polygons and the options are circle, rectangle or exit. Then we will ask the user to enter the value for variable polygon input, choose the polygon type or exit. Based on the input given by the user, we will check if polygon is equal, equal to circle then do something else if polygon is equal, equal to rectangle, again do something else if polygon is equal, equal to exit. In this case, we will break this particular while loop and terminate this Python code.

And the last option is else, which says please select the correct polygon type or exit. This particular else block is required because user may enter a wrong input for this particular statement, which is not a part of the given list of inputs, which means we are asking user to choose only one option out of these three given options, which are circle, rectangle or exit. In case, if user selects a different option which is not available in the given list, then this particular else block will execute which will indicate the user that the given input is incorrect and the correct input has to be provided in order to execute the program.

Now, the next step is to write a Python code for polygon is equal to circle, which means if the user selects polygon as a circle, then we have to move to the next level of menu where we are supposed to ask a different properties of a circle, a menu which has two different options, area and perimeter. Now, let us add that particular code block inside this if.

As a next step, first we will declare one variable called property and initialize it with a empty string. Just like a polygon variable, this property variable will be used to accept input from user which can be either area or perimeter. Then we will accept the value for radius. Once that value is there, we will execute this particular while loop till the value of property is empty. This particular loop will print the second level of menu which says circle, properties, first option area, second option perimeter or we can go back to previous menu.

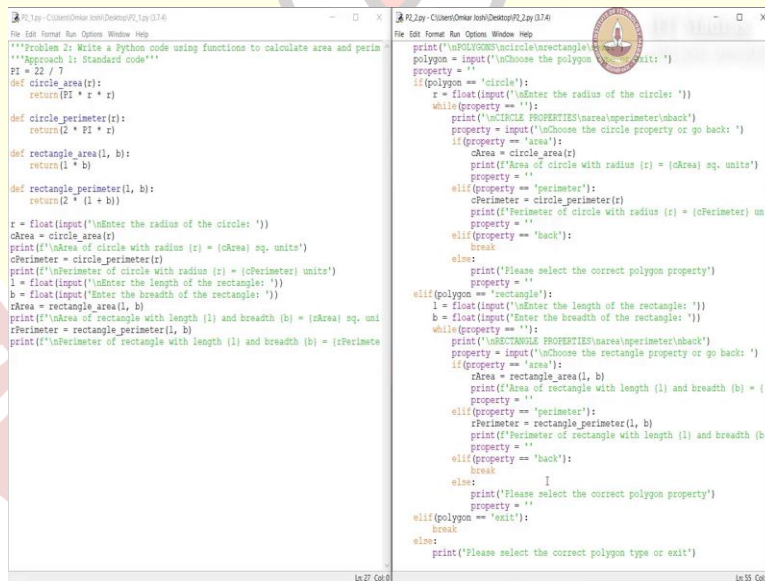
Now, we will ask the user to select one option from this second level of menu which is circle properties. If the user selects area, then we will do something. If user enters perimter as a

property, then we should do something. If user selects property as back, which means we should terminate this inner while loop which is printing second level of properties and we must go back to original while loop which is printing first level of menu for polygons.

Once again, a else block, in case if user enters option which is not available in the given menu which will print indication message to the user which says please select the correct polygon property and we will reset the value of variable property back to empty string. Now, we know what we are supposed to do when a property is equal, equal to area. In this case, we should do exactly what we did earlier, which is to call this particular function circle underscore area and then we should print the result.

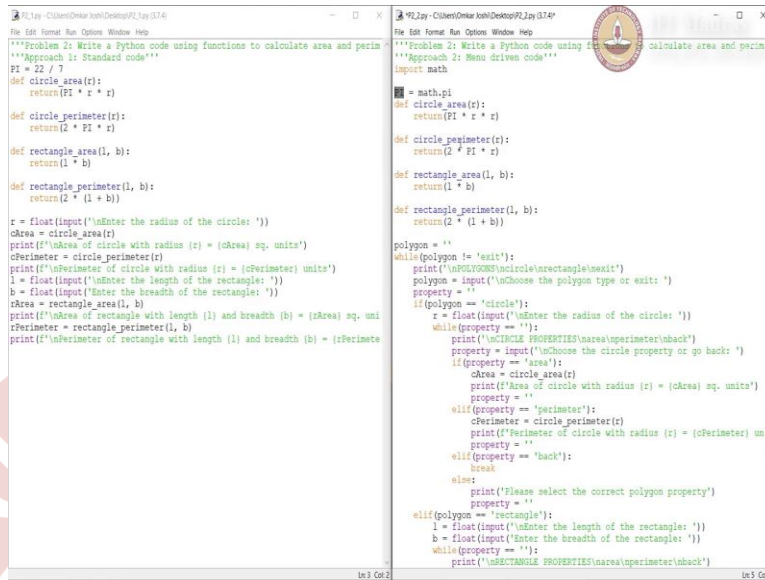
But in addition to that, once again, we should reset the value of property to empty string in order to accept the next input. Similarly, if the user selects property is equal, equal to perimeter, we should execute this particular function circle underscore perimeter and print the result along with resetting this particular variable property.

(Refer Slide Time: 28:14)



```
File Edit Format Run Options Window Help
***Problem 2: Write a Python code using functions to calculate area and perim
Approach 1: Standard code***
PI = 22 / 7
def circle_area(r):
    return (PI * r * r)
def circle_perimeter(r):
    return (2 * PI * r)
def rectangle_area(l, b):
    return (l * b)
def rectangle_perimeter(l, b):
    return (2 * (l + b))
r = float(input("Enter the radius of the circle: "))
cArea = circle_area(r)
print(f"Area of circle with radius (r) = {cArea} sq. units")
cPerimeter = circle_perimeter(r)
print(f"Perimeter of circle with radius (r) = {cPerimeter} units")
l = float(input("Enter the length of the rectangle: "))
b = float(input("Enter the breadth of the rectangle: "))
rArea = rectangle_area(l, b)
print(f"Area of rectangle with length (l) and breadth (b) = {rArea} sq. uni
rPerimeter = rectangle_perimeter(l, b)
print(f"Perimeter of rectangle with length (l) and breadth (b) = {rPerime

File Edit Format Run Options Window Help
print("\nPOLYGONS\n1.circle\n2.rectangle\n3.exit: ")
polygon = input("\nChoose the polygon to calculate: ")
property = ""
if (polygon == "circle"):
    r = float(input("Enter the radius of the circle: "))
    while (property == ""):
        print("\n1.CIRCLE\n2.AREA\n3.PERIMETER\n4.BACK: ")
        property = input("\nChoose the circle property or go back: ")
        if (property == "area"):
            cArea = circle_area(r)
            print(f"Area of circle with radius (r) = {cArea} sq. units")
            property = ""
        elif (property == "perimeter"):
            cPerimeter = circle_perimeter(r)
            print(f"Perimeter of circle with radius (r) = {cPerimeter} un
            property = ""
        elif (property == "back"):
            break
    else:
        print("Please select the correct polygon property")
        property = ""
elif (polygon == "rectangle"):
    l = float(input("Enter the length of the rectangle: "))
    b = float(input("Enter the breadth of the rectangle: "))
    while (property == ""):
        print("\n1.RECTANGLE\n2.AREA\n3.PERIMETER\n4.BACK: ")
        property = input("\nChoose the rectangle property or go back: ")
        if (property == "area"):
            rArea = rectangle_area(l, b)
            print(f"Area of rectangle with length (l) and breadth (b) = {
            property = ""
        elif (property == "perimeter"):
            rPerimeter = rectangle_perimeter(l, b)
            print(f"Perimeter of rectangle with length (l) and breadth (b
            property = ""
        elif (property == "back"):
            break
    else:
        print("Please select the correct polygon property")
        property = ""
elif (polygon == "exit"):
    break
else:
    print("Please select the correct polygon type or exit")
```



```
'''Problem 2: Write a Python code using functions to calculate area and perim
'''
'''Approach 1: Standard code'''
PI = 22 / 7

def circle_area(r):
    return(PI * r * r)

def circle_perimeter(r):
    return(2 * PI * r)

def rectangle_area(l, b):
    return(l * b)

def rectangle_perimeter(l, b):
    return(2 * (l + b))

r = float(input("Enter the radius of the circle: "))
cArea = circle_area(r)
print(f"Area of circle with radius (r) = {cArea} sq. units")
cPerimeter = circle_perimeter(r)
print(f"Perimeter of circle with radius (r) = {cPerimeter} units")
l = float(input("Enter the length of the rectangle: "))
b = float(input("Enter the breadth of the rectangle: "))
rArea = rectangle_area(l, b)
print(f"Area of rectangle with length (l) and breadth (b) = {rArea} sq. uni
rPerimeter = rectangle_perimeter(l, b)
print(f"Perimeter of rectangle with length (l) and breadth (b) = {rPerimete

'''Problem 2: Write a Python code using functions to calculate area and perim
'''
'''Approach 2: Menu driven code'''
import math

PI = math.pi

def circle_area(r):
    return(PI * r * r)

def circle_perimeter(r):
    return(2 * PI * r)

def rectangle_area(l, b):
    return(l * b)

def rectangle_perimeter(l, b):
    return(2 * (l + b))

polygon = ''
while(polygon != 'exit'):
    print("\nPOLYGON(circle/rectangle/exit)")
    polygon = input("Choose the polygon type or exit: ")
    property = ''
    if(polygon == 'circle'):
        r = float(input("Enter the radius of the circle: "))
        while(property != ''):
            print("\nCIRCLE PROPERTIES\narea/perimeter/back")
            property = input("Choose the circle property or go back: ")
            if(property == 'area'):
                cArea = circle_area(r)
                print(f"Area of circle with radius (r) = {cArea} sq. units")
                property = ''
            elif(property == 'perimeter'):
                cPerimeter = circle_perimeter(r)
                print(f"Perimeter of circle with radius (r) = {cPerimeter} un
                property = ''
            elif(property == 'back'):
                break
            else:
                print("Please select the correct polygon property")
                property = ''
    elif(polygon == 'rectangle'):
        l = float(input("Enter the length of the rectangle: "))
        b = float(input("Enter the breadth of the rectangle: "))
        while(property != ''):
            print("\nRECTANGLE PROPERTIES\narea/perimeter/back")
```

Now, as our sub-menu for circle properties is ready, we should create similar sub-menu for rectangle properties as well, else if polygon is equal, equal to rectangle accept length, breadth, then till property is equal, equal to empty, we will print this particular menu rectangle, properties, first option area, second option perimeter, third option back then we will indicate the user to enter the input. If the input is area, we will call this function rectangle underscore area exactly like what we did over here and then we will print the result.

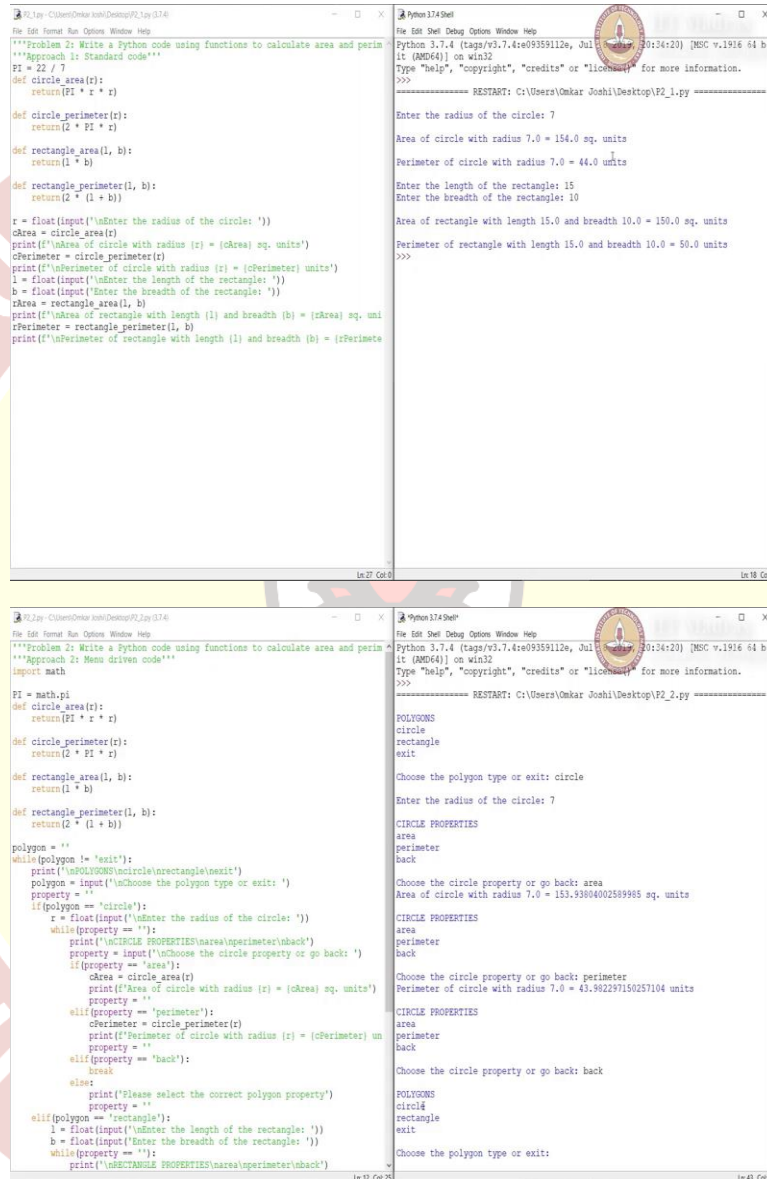
Similarly, we will do it for perimeter of the rectangle. In this case also we have to reset the value of variable property every time in order to execute this particular while loop which is printing this second level of menu which says rectangle properties. Then, if user selects back option, we will terminate the inner while loop. Else we will indicate the user to select the correct polygon property. This is how a standard program can be converted into a menu driven code, where the main logic which is written in the functions stays as it is. Only change happens with respect to various print statement and input statements.

In this code, we have not yet declared this particular variable pi as 22 by 7, because as we have improved the code from approach one to approach two, let us try to improve the accuracy of variable pi as well, which can be achieved using an inbuilt library called math. This particular math library has predefined the value of variable pi, which can be used using math dot pi. Now, we will use this updated value of pi which will give us the more accurate results with respect to area as well as perimeter.



Now, let us try to execute both the programs, first using the standard approach, then using the menu driven code.

(Refer Slide Time: 30:57)



```
File Edit Format Run Options Window Help
***Problem 2: Write a Python code using functions to calculate area and perim
***Approach 1: Standard code***
PI = 22 / 7
def circle_area(r):
    return (PI * r * r)

def circle_perimeter(r):
    return (2 * PI * r)

def rectangle_area(l, b):
    return (l * b)

def rectangle_perimeter(l, b):
    return (2 * (l + b))

r = float(input("Enter the radius of the circle: "))
cArea = circle_area(r)
print(f"Area of circle with radius (r) = {cArea} sq. units")
cPerimeter = circle_perimeter(r)
print(f"Perimeter of circle with radius (r) = {cPerimeter} units")
l = float(input("Enter the length of the rectangle: "))
b = float(input("Enter the breadth of the rectangle: "))
rArea = rectangle_area(l, b)
print(f"Area of rectangle with length (l) and breadth (b) = {rArea} sq. uni
rPerimeter = rectangle_perimeter(l, b)
print(f"Perimeter of rectangle with length (l) and breadth (b) = {rPerimeter}

Python 3.7.4 Shell
Python 3.7.4 (tags/v3.7.4:ec09359112e, Jul 8, 2019; 30:34:20) [MSC v.1916 64 b
it (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Omkar Joshi\Desktop\P2_1.py =====>>>
Enter the radius of the circle: 7
Area of circle with radius 7.0 = 154.0 sq. units
Perimeter of circle with radius 7.0 = 44.0 units
Enter the length of the rectangle: 15
Enter the breadth of the rectangle: 10
Area of rectangle with length 15.0 and breadth 10.0 = 150.0 sq. units
Perimeter of rectangle with length 15.0 and breadth 10.0 = 50.0 units
>>>

File Edit Format Run Options Window Help
***Problem 2: Write a Python code using functions to calculate area and perim
***Approach 2: Menu driven code***
import math
PI = math.pi
def circle_area(r):
    return (PI * r * r)

def circle_perimeter(r):
    return (2 * PI * r)

def rectangle_area(l, b):
    return (l * b)

def rectangle_perimeter(l, b):
    return (2 * (l + b))

polygon = ""
while (polygon != "exit"):
    print("\nPOLYGONS\n(circle)\n(rectangle)\n(exit)")
    polygon = input("\nChoose the polygon type or exit: ")
    property = ""
    if (polygon == "circle"):
        r = float(input("\nEnter the radius of the circle: "))
        while (property == ""):
            print("\nCIRCLE PROPERTIES\narea\nperimeter\nback")
            property = input("\nChoose the circle property or go back: ")
            if (property == "area"):
                cArea = circle_area(r)
                print(f"Area of circle with radius (r) = {cArea} sq. units")
            elif (property == "perimeter"):
                cPerimeter = circle_perimeter(r)
                print(f"Perimeter of circle with radius (r) = {cPerimeter} un
            elif (property == "back"):
                break
        else:
            print("Please select the correct polygon property")
            property = ""
    elif (polygon == "rectangle"):
        l = float(input("\nEnter the length of the rectangle: "))
        b = float(input("\nEnter the breadth of the rectangle: "))
        while (property == ""):
            print("\nRECTANGLE PROPERTIES\narea\nperimeter\nback")

Python 3.7.4 Shell
Python 3.7.4 (tags/v3.7.4:ec09359112e, Jul 8, 2019; 30:34:20) [MSC v.1916 64 b
it (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Omkar Joshi\Desktop\P2_2.py =====>>>
POLYGONS
circle
rectangle
exit
Choose the polygon type or exit: circle
Enter the radius of the circle: 7
CIRCLE PROPERTIES
area
perimeter
back
Choose the circle property or go back: area
Area of circle with radius 7.0 = 153.93804002589965 sq. units
CIRCLE PROPERTIES
area
perimeter
back
Choose the circle property or go back: perimeter
Perimeter of circle with radius 7.0 = 43.982297150257104 units
CIRCLE PROPERTIES
area
perimeter
back
Choose the circle property or go back: back
Choose the circle property or go back: back
POLYGONS
circle
rectangle
exit
Choose the polygon type or exit:
>>>
```

```

'''Problem 2: Write a Python code using functions to calculate area and perim
'''
'''Approach 2: Menu driven code'''
import math

PI = math.pi
def circle_area(r):
    return(PI * r * r)

def circle_perimeter(r):
    return(2 * PI * r)

def rectangle_area(l, b):
    return(l * b)

def rectangle_perimeter(l, b):
    return(2 * (l + b))

polygon = ''
while(polygon != 'exit'):
    print("\nPOLYGONS(circle/rectangle/exit)")
    polygon = input("\nChoose the polygon type or exit: ")
    property = ''
    if(polygon == 'circle'):
        r = float(input("\nEnter the radius of the circle: "))
        while(property == ''):
            print("\nRECTANGLE PROPERTIES\narea/perimeter/back")
            property = input("\nChoose the circle property or go back: ")
            if(property == 'area'):
                area = circle_area(r)
                print("Area of circle with radius (r) = (k)area sq. units")
                property = ''
            elif(property == 'perimeter'):
                perimeter = circle_perimeter(r)
                print("Perimeter of circle with radius (r) = (k)perimeter")
                property = ''
            elif(property == 'back'):
                break
        else:
            print("Please select the correct polygon property")
    elif(polygon == 'rectangle'):
        l = float(input("\nEnter the length of the rectangle: "))
        b = float(input("\nEnter the breadth of the rectangle: "))
        while(property == ''):
            print("\nRECTANGLE PROPERTIES\narea/perimeter/back")

```

Enter the radius of circle 7, area of circle with radius 7.0 equal to 154 square units which is correct, perimeter of circle with radius 7.0 is equal to 44.0 units which is again correct, enter the length of rectangle 15, breadth 10, hence the area is 150 square units and perimeter is 50 units. All these numbers are current, but over here as a user we are not getting option to decide whether we want only area or only perimeter or only properties of a circle or a rectangle. All these different options are not available with us.

In order to achieve that, we will move to second approach which is a menu driven code. First level menu, polygons, circle, rectangle, exit. Let us choose circle. Then it will ask us for the radius of the circle. Once the radius of circle is in place, then the second level of menu will be printed which has circle properties, first area, second perimter or we can go back. Let us try area first.

Now, if you look at this particular value, we are getting more accurate value because we have used this particular math library to initialize variable pi. After this particular output, you can observe that the same menu is getting printed once again and the program is asking us to select the circle property once again or we can go back.

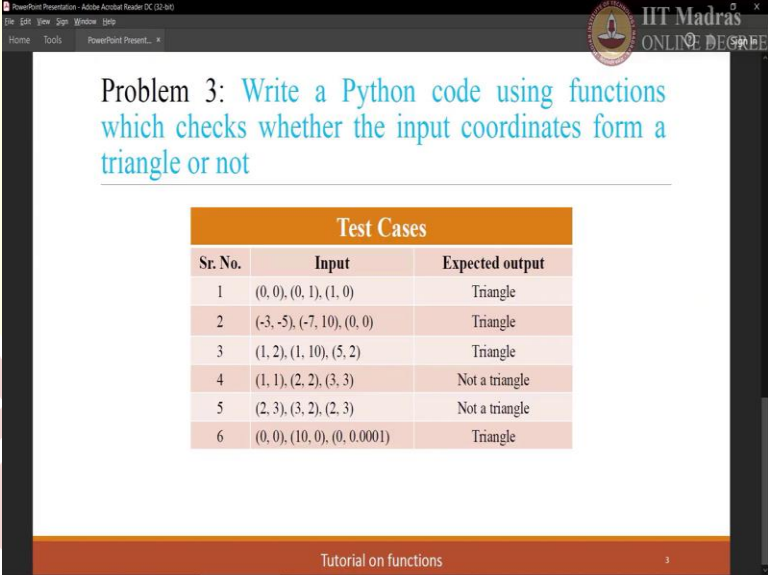
Let us try for perimeter. Now to check the properties of rectangle, we can go back from here which will display the first level of menu which has different polygon options, circle, rectangle or we can exit. Let us choose a rectangle, enter the length of a triangle 15, breadth 10. After this

it will print a second level of menu, but this time it will be a rectangle properties, because we selected the rectangle as option from the first level of menu.

Let us choose area, it will display the value of area then perimeter. This particular approach allows us the flexibility to decide which specific polygon we want to select. On top of that, we can also decide which specific property we want to look at with respect to that particular polygon. Whenever we are done with a specific polygon, we can always go back and explore a different polygon or we also have an option to calculate area or a perimeter of a circle or rectangle again and again with different radius or different values for length and breadth, because the program will keep executing till we enter a specific word which is exit.

Let us try that option as well. We will go back from here. Now we are in first level of menu. Let us try exit. The program is terminated. This added flexibility for a user is the main reason to write a menu driven program.

(Refer Slide Time: 35:26)



Problem 3: Write a Python code using functions which checks whether the input coordinates form a triangle or not

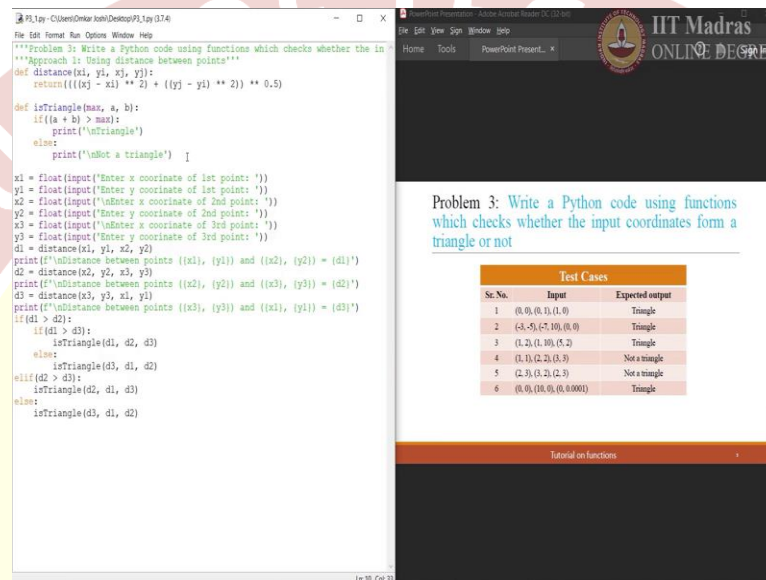
Test Cases		
Sr. No.	Input	Expected output
1	(0, 0), (0, 1), (1, 0)	Triangle
2	(-3, -5), (-7, 10), (0, 0)	Triangle
3	(1, 2), (1, 10), (5, 2)	Triangle
4	(1, 1), (2, 2), (3, 3)	Not a triangle
5	(2, 3), (3, 2), (2, 3)	Not a triangle
6	(0, 0), (10, 0), (0, 0.0001)	Triangle

Tutorial on functions 3

Let us move to next problem statement, problem statement three. Write a Python code using functions which checks whether the input coordinates form a triangle are not. In the given test cases, the x and y coordinates of each point are given to us. Based on the coordinates of these three points, we have to decide whether they will form a triangle or not. Mathematically, there are many ways to identify whether the given points can form a triangle or not.

For the purpose of demonstration of this Python code, we will write a code using two different approaches. First approach, we will use the distance between the points. Second approach will focus on slope of lines drawn using the given points. Let us look at the Python code using the first approach.

(Refer Slide Time: 36:31)



```

File Edit Format Run Options Window Help
***Problem 3: Write a Python code using functions which checks whether the in
***Approach 1: Using distance between points***
def distance(x1, y1, x2, y2):
    return(((x2 - x1) ** 2) + ((y2 - y1) ** 2)) ** 0.5

def isTriangle(a, b, c):
    if((a + b) > c):
        print("isTriangle")
    else:
        print("isNot a triangle") ]

x1 = float(input("Enter x coordinate of 1st point: "))
y1 = float(input("Enter y coordinate of 1st point: "))
x2 = float(input("Enter x coordinate of 2nd point: "))
y2 = float(input("Enter y coordinate of 2nd point: "))
x3 = float(input("Enter x coordinate of 3rd point: "))
y3 = float(input("Enter y coordinate of 3rd point: "))

d1 = distance(x1, y1, x2, y2)
print(f"Distance between points ((x1), (y1)) and ((x2), (y2)) = (d1)")
d2 = distance(x2, y2, x3, y3)
print(f"Distance between points ((x2), (y2)) and ((x3), (y3)) = (d2)")
d3 = distance(x3, y3, x1, y1)
print(f"Distance between points ((x3), (y3)) and ((x1), (y1)) = (d3)")

if(d1 > d2):
    if(d1 > d3):
        isTriangle(d1, d2, d3)
    else:
        isTriangle(d3, d1, d2)
elif(d2 > d3):
    isTriangle(d2, d1, d3)
else:
    isTriangle(d3, d1, d2)

```

Problem 3: Write a Python code using functions which checks whether the input coordinates form a triangle or not

Test Cases		
Sl. No.	Input	Expected output
1	(0, 0), (0, 1), (1, 0)	Triangle
2	(-1, -5), (-7, 10), (0, 0)	Triangle
3	(1, 2), (1, 10), (5, 2)	Triangle
4	(1, 1), (2, 2), (3, 3)	Not a triangle
5	(2, 3), (3, 2), (2, 3)	Not a triangle
6	(0, 0), (10, 0), (0, 0.00001)	Triangle

Tutorial on functions

First, we have to accept these x and y coordinates for three different points, which we will do using a variable x1, y1, x2, y2, x3, y3 using float of input of into x or y coordinate of first, second or third point. Once we have coordinates for all three points, then the next step is to find a distance between each pair of coordinates given to us. We have to calculate distance between coordinate 1 and 2, then 2 and 3 and at the end between 1 and 3, which means we have to execute same logic three different times to calculate the distance among these points, which motivates us to write a function called distance, which can be called three different times using the different parameters.

For example, for first distance d1, we will pass parameters x1, y1, x2, y2. Similarly, for second distance, we can pass parameters x2, y2 and x3, y3, and then for third distance, x3, y3, x1, y1. Now, as we have decided that we will create a function called distance, which will calculate the distance between the given two coordinates and return the value back which can be stored into variables d1, d2 and d3. Let us look at that particular function which calculates the distance.

As we all know, the distance between two coordinates can be calculated using this particular formula. As we have studied earlier, this particular operator is used to calculate the exponential value of a given number. First, subtraction will happen, after that we will do a square of that result. Similar will happen on this side. After that, we will do a summation of those two computed results then we are doing power of 0.5. Once all three distances are calculated, then we will use the triangle property which says, if sum of two distances is greater than third distance, then it is a triangle.

But we do not have to check this property for all different combinations. Because, if we check that, the sum of two smallest distances is greater than the other distance, then definitely it will be a triangle. In order to do that, first we have to compare these three distances to find which distance is the highest one which can be done using this kind of a logic, where if  $d_1$  is greater than  $d_2$  and  $d_1$  is greater than  $d_3$  which means  $d_1$  is the highest one or else  $d_3$  is the highest, else if  $d_2$  is greater than  $d_3$  then  $d_2$  is the highest distance else  $d_3$  is the highest distance.

Once we know which distance is the highest distance, then we can write one function called `isTriangle` which will find whether  $d_2$  plus  $d_3$  is greater than  $d_1$  or not. Let us try to write that function. This particular function `isTriangle` will accept three different distances. First parameter will hold the highest distance, whereas second and third parameter will hold the remaining two distances. As we seen earlier, if sum of two smaller distances is greater than the longest distance then definitely the given coordinates will form a triangle. Let us test this particular code against the given inputs.



(Refer Slide Time: 41:24)

The image displays two screenshots side-by-side. The left screenshot shows a Python 3.7.4 shell window with a script that calculates the distance between three points and checks if they form a triangle. The right screenshot shows a presentation slide from IIT Madras with the same problem statement and test cases.

**Python Shell Output:**

```
Python 3.7.4 (tags/v3.7.4:0935112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 b...
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Omkar Joshi\Desktop\PJ_1.py =====
Enter x coordinate of 1st point: 0
Enter y coordinate of 1st point: 0

Enter x coordinate of 2nd point: 0
Enter y coordinate of 2nd point: 1

Enter x coordinate of 3rd point: 1
Enter y coordinate of 3rd point: 0

Distance between points (0.0, 0.0) and (0.0, 1.0) = 1.0
Distance between points (0.0, 1.0) and (1.0, 0.0) = 1.4142135623730951
Distance between points (1.0, 0.0) and (0.0, 0.0) = 1.0

Triangle
>>>
===== RESTART: C:\Users\Omkar Joshi\Desktop\PJ_1.py =====
Enter x coordinate of 1st point: -3
Enter y coordinate of 1st point: -5

Enter x coordinate of 2nd point: -7
Enter y coordinate of 2nd point: 10

Enter x coordinate of 3rd point: 0
Enter y coordinate of 3rd point: 0

Distance between points (-3.0, -5.0) and (-7.0, 10.0) = 15.52417469260024
Distance between points (-7.0, 10.0) and (0.0, 0.0) = 12.206555615733702
Distance between points (0.0, 0.0) and (-3.0, -5.0) = 5.830951894945301

Triangle
>>>
===== RESTART: C:\Users\Omkar Joshi\Desktop\PJ_1.py =====
Enter x coordinate of 1st point: 1
Enter y coordinate of 1st point: 2

Enter x coordinate of 2nd point: 1
Enter y coordinate of 2nd point: 2

Distance between points (1.0, 2.0) and (1.0, 10.0) = 8.0
Distance between points (1.0, 10.0) and (5.0, 2.0) = 8.94427190999916
Distance between points (5.0, 2.0) and (1.0, 2.0) = 4.0

Triangle
>>>
===== RESTART: C:\Users\Omkar Joshi\Desktop\PJ_1.py =====
Enter x coordinate of 1st point: 1
Enter y coordinate of 1st point: 1

Enter x coordinate of 2nd point: 2
Enter y coordinate of 2nd point: 2

Enter x coordinate of 3rd point: 3
Enter y coordinate of 3rd point: 3

Distance between points (1.0, 1.0) and (2.0, 2.0) = 1.4142135623730951
Distance between points (2.0, 2.0) and (3.0, 3.0) = 1.4142135623730951
Distance between points (3.0, 3.0) and (1.0, 1.0) = 2.8284271247461903

Not a triangle
>>>
===== RESTART: C:\Users\Omkar Joshi\Desktop\PJ_1.py =====
Enter x coordinate of 1st point: 2
Enter y coordinate of 1st point: 3

Enter x coordinate of 2nd point: 3
Enter y coordinate of 2nd point: 2

Enter x coordinate of 3rd point: 2
Enter y coordinate of 3rd point: 3

Distance between points (2.0, 3.0) and (3.0, 2.0) = 1.4142135623730951
Distance between points (3.0, 2.0) and (2.0, 3.0) = 1.4142135623730951
Distance between points (2.0, 3.0) and (2.0, 3.0) = 0.0

Not a triangle
>>>
```

**Presentation Slide:**

**Problem 3: Write a Python code using functions which checks whether the input coordinates form a triangle or not**

Sr. No.	Input	Expected output
1	(0, 0), (0, 1), (1, 0)	Triangle
2	(-4, -5), (-7, 10), (0, 0)	Triangle
3	(1, 2), (1, 10), (5, 2)	Triangle
4	(1, 1), (2, 2), (3, 3)	Not a triangle
5	(2, 3), (3, 2), (2, 3)	Not a triangle
6	(0, 0), (0, 0), (0, 0.0001)	Triangle

Enter x coordinate of first point 0, enter y coordinate of first point 0, enter x coordinate of second point 0, enter y coordinate of second point 1, enter x coordinate of third point 1, y coordinate of third point 0. We got the distance between all these three points and we can see that it forms a triangle. Let us execute the code using remaining test cases, minus 3, minus 5, minus 7, 10, 0, 0. Again it will form a triangle, 1, 2, 1, 10, 5, 2, once again it will form a triangle. Fourth test case, 1, 1, 2, 2, 3, 3, not a triangle. Fifth test case, 2, 3, 3, 2, 2, 3, once again not triangle, because here a same coordinate is given twice, which means we will get a line not a triangle.

(Refer Slide Time: 43:12)

The collage consists of four images arranged in a 2x2 grid. The top-left image is a screenshot of a Python 3.7.4 shell window. It shows the user entering coordinates for three points: (0,0), (10,0), and (0,0.0001). The shell calculates the distances between these points and then uses Heron's formula to calculate the area of the triangle, which is 10.0000000005. The top-right image is a screenshot of a PowerPoint presentation slide titled 'Problem 3: Write a Python code using functions which checks whether the input coordinates form a triangle or not'. It includes a table of test cases with 6 rows and 3 columns: S.No., Input, and Expected output. The bottom-left image is a screenshot of a Python script file named 'P3\_2ay.py'. It defines a function 'slope(x1, y1, x2, y2)' that returns the slope of the line connecting two points. The script then uses this function to calculate the slopes of the three sides of the triangle and checks if they are equal to determine if it is a triangle. The bottom-right image is another screenshot of the same PowerPoint slide, showing the test cases table again.

Python 3.7.4 Shell

```
Python 3.7.4 (tags/v3.7.4:0935112e, 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\P3_1.py =====
Enter x coordinate of 1st point: 0
Enter y coordinate of 1st point: 0

Enter x coordinate of 2nd point: 10
Enter y coordinate of 2nd point: 0

Enter x coordinate of 3rd point: 0
Enter y coordinate of 3rd point: 0.0001

Distance between points (0.0, 0.0) and (10.0, 0.0) = 10.0
Distance between points (10.0, 0.0) and (0.0, 0.0001) = 10.0000000005
Distance between points (0.0, 0.0001) and (0.0, 0.0) = 0.0001

Triangle
>>> |
```

Problem 3: Write a Python code using functions which checks whether the input coordinates form a triangle or not

Test Cases		
Sr. No.	Input	Expected output
1	(0, 0), (0, 1), (1, 0)	Triangle
2	(-4, -5), (-7, 10), (0, 0)	Triangle
3	(1, 2), (1, 10), (5, 2)	Triangle
4	(1, 1), (2, 2), (3, 3)	Not a triangle
5	(2, 3), (3, 2), (2, 3)	Not a triangle
6	(0, 0), (10, 0), (0, 0.0001)	Triangle

Tutorial on functions

13.11 x 7.50 m

Python 3.7.4 Shell

```
***Problem 3: Write a Python code using functions which checks whether the in
***Approach 2: Using slope of lines connecting two points***
import math

def slope(x1, y1, x2, y2):
    if(x1 == x2):
        return(math.inf)
    else:
        return((y2 - y1) / (x2 - x1))

x1 = float(input("Enter x coordinate of 1st point: "))
y1 = float(input("Enter y coordinate of 1st point: "))
x2 = float(input("Enter x coordinate of 2nd point: "))
y2 = float(input("Enter y coordinate of 2nd point: "))
x3 = float(input("Enter x coordinate of 3rd point: "))
y3 = float(input("Enter y coordinate of 3rd point: "))
s1 = slope(x1, y1, x2, y2)
print(f"Slope of the line connecting points ((x1), (y1)) and ((x2), (y2)) =
s2 = slope(x2, y2, x3, y3)
print(f"Slope of the line connecting points ((x2), (y2)) and ((x3), (y3)) =
if(s1 == s2):
    print("isTriangle")
else:
    print("isNot a triangle") |
```

Problem 3: Write a Python code using functions which checks whether the input coordinates form a triangle or not

Test Cases		
Sr. No.	Input	Expected output
1	(0, 0), (0, 1), (1, 0)	Triangle
2	(-4, -5), (-7, 10), (0, 0)	Triangle
3	(1, 2), (1, 10), (5, 2)	Triangle
4	(1, 1), (2, 2), (3, 3)	Not a triangle
5	(2, 3), (3, 2), (2, 3)	Not a triangle
6	(0, 0), (10, 0), (0, 0.0001)	Triangle

Tutorial on functions

13.11 x 7.50 m

Last test case, 0, 0, 10, 0, 0, 0, 0, 0, 0, 1. Once again, it will form a triangle. Now, let us look at the second approach which uses the slope. Once again we will start by accepting x and y coordinates of all three points. Once these coordinates are available we will try to find a slope between the given coordinates. In this case as well, we can draw three different lines. Hence, we have to calculate slope of three different lines, which will again motivate the use of function to calculate a slope of a line. Slope s1 is equal to slope between x1, y1 and x2, y2, similarly, slope s2 between x2, y2 and x3, y3 and then slope s3 is equal to slope x3, y3, x1, y1.

Next step is to write this particular function slope. As we know, the formula to find a slope of a line is  $y_j$  minus  $y_i$  divided by  $x_j$  minus  $x_i$ , if  $x_j$  is not equal to  $x_i$ . But if  $x_i$  is equal to  $x_j$  which will make denominator 0, then we can use this math library once again which has a way to return infinity using `math dot inf`. Once this particular function executes using three different combinations, we will get slope  $s_1$ ,  $s_2$  and  $s_3$ . But if you think carefully, we will realize that if a slope of any two lines is equal then it cannot be a triangle.

Hence, we do not even require the value of third slope to figure out whether it will form a triangle or not. Hence, we can remove these two lines and check whether  $s_1$  is equal to  $s_2$  or not. If  $s_1$  not equal to  $s_2$  then it must be a triangle, otherwise it will not be a triangle. Let us try this Python code with the given inputs.

(Refer Slide Time: 46:23)

The Python 3.7.4 shell window displays the following code and output:

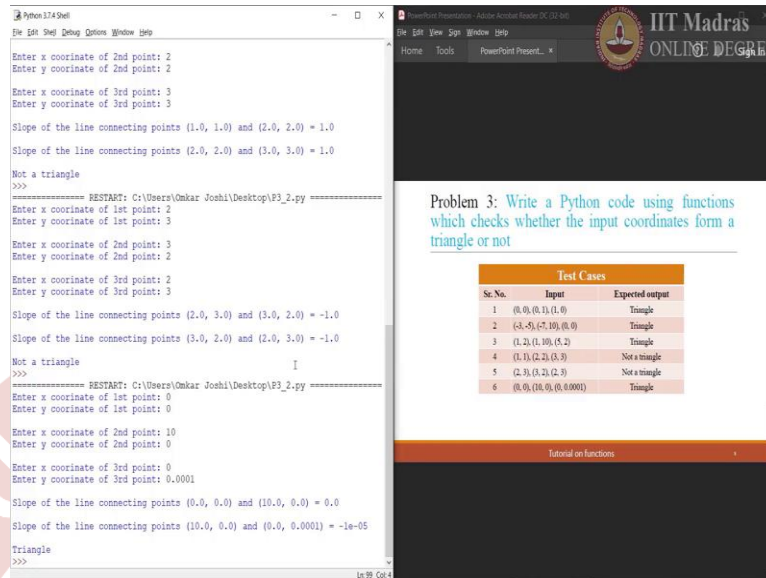
```
Python 3.7.4 (tags/v3.7.4:0e939112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Omkar Joshi\Desktop\p3_2.py =====
Enter x coordinate of 1st point: 0
Enter y coordinate of 1st point: 0
Enter x coordinate of 2nd point: 0
Enter y coordinate of 2nd point: 1
Enter x coordinate of 3rd point: 1
Enter y coordinate of 3rd point: 0
Slope of the line connecting points (0.0, 0.0) and (0.0, 1.0) = inf
Slope of the line connecting points (0.0, 1.0) and (1.0, 0.0) = -1.0
Triangle
>>>
===== RESTART: C:\Users\Omkar Joshi\Desktop\p3_2.py =====
Enter x coordinate of 1st point: -3
Enter y coordinate of 1st point: -5
Enter x coordinate of 2nd point: -7
Enter y coordinate of 2nd point: 10
Enter x coordinate of 3rd point: 0
Enter y coordinate of 3rd point: 0
Slope of the line connecting points (-3.0, -5.0) and (-7.0, 10.0) = -3.75
Slope of the line connecting points (-7.0, 10.0) and (0.0, 0.0) = -1.4285714285714286
Triangle
>>>
===== RESTART: C:\Users\Omkar Joshi\Desktop\p3_2.py =====
Enter x coordinate of 1st point: 1
Enter y coordinate of 1st point: 2
Enter x coordinate of 2nd point: 1
Enter y coordinate of 2nd point: 10
```

The PowerPoint slide, titled "IIT Madras ONLINE DEGREE", contains the following text:

Problem 3: Write a Python code using functions which checks whether the input coordinates form a triangle or not

Test Cases		
Sr. No.	Input	Expected output
1	(0, 0), (0, 1), (1, 0)	Triangle
2	(-3, -5), (-7, 10), (0, 0)	Triangle
3	(1, 2), (1, 10), (5, 2)	Triangle
4	(1, 1), (2, 2), (3, 3)	Not a triangle
5	(2, 3), (3, 2), (2, 3)	Not a triangle
6	(0, 0), (10, 0), (0, 0.00001)	Triangle

Below the table, it says "Tutorial on functions".



Let us try this Python code with the given test cases, 0, 0, 0, 1, 1, 0, it is a triangle, minus 3, minus 5, minus 7, 10, 0, 0, once again it is a triangle, 1, 2, 1, 10, 5, 2, triangle, 1, 1, 2, 2, 3, 3 not a triangle, 2, 3, 3, 2, 2, 3 not a triangle, 0, 0, 10, 0, 0, 0.0001, again triangle. As we can see, both these approaches are given the expected output. It depends on the programmer to decide which approach to choose in order to achieve the given problem statement. With respect to both these approaches, we can say that these two python codes has same interface but different implementation.

Now based on the Python code which we saw in these three problem statements, we can comfortably say that writing some part of the code as a function was very helpful, because in this case we had to calculate distance three different times. We avoided writing same code thrice by putting the common code into a function. Similarly, we avoided writing same code to calculate slope twice because we added that common code into a function.

If you go back to previous question, in this problem statement functions help us in two different ways. First, it allowed us to create a menu driven code and second it also gave us a flexibility to always add more polygons or more properties of the polygons. For example, if we decide to add a 3D polygon, then we can always add a surface area or a volume and so on. Similarly, in first problem statement, we can always add some more properties for the sentence like how many vowels are there in the sentence or how many special characters or how many sentences are there

in the given input and so on. Hence, we can conclude that function allows us to update the code without affecting the existing program.

Thank you for watching this tutorial. Happy learning.

