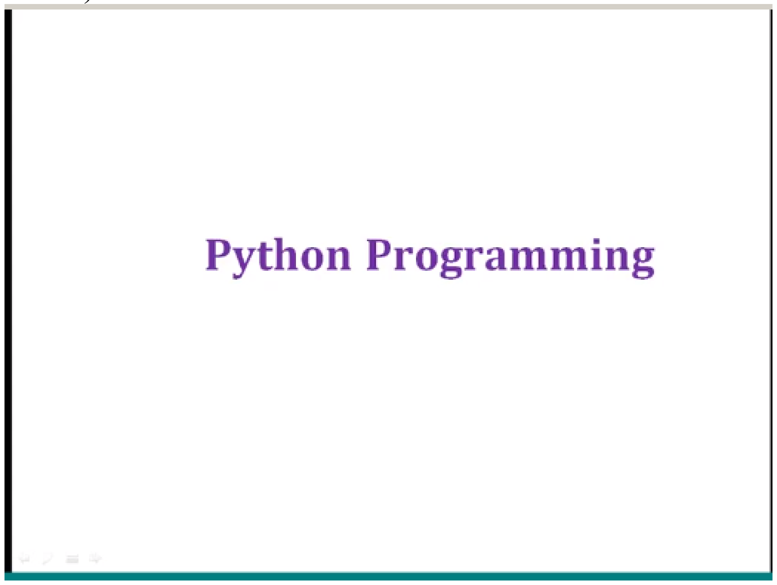
(Refer Slide Time: 00:00)



Python Programming

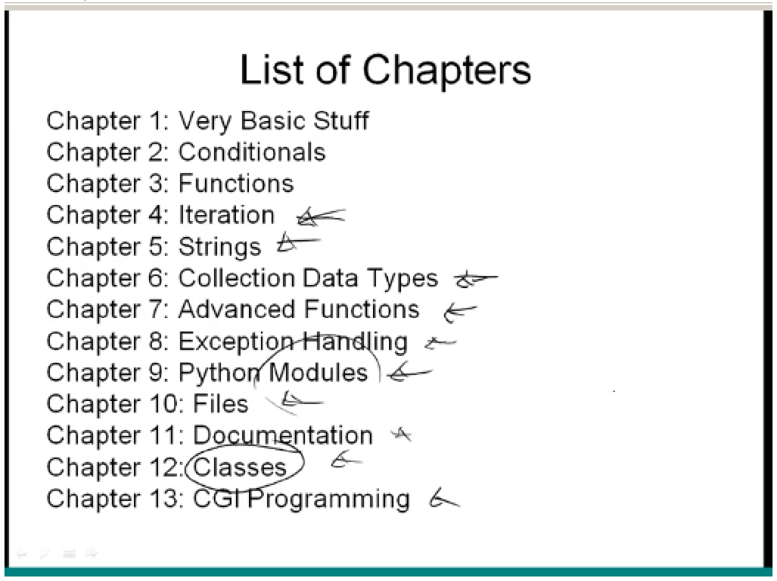
Hi everyone this is the lower the programming class essentially we covered several topics, until now the first started with the UNIX system talked about the ins and outs of UNIX, went through the UNIX networking, then we looked at Perl programming in detail we went through all the aspects of Perl, how the various variable specification what types of variables that so will have, then we went through like the data structure essentially and then we also looked at the control structure.

And then now we got proficient enough in Perl then we started looking at TCL -TCL and TK initially we started with TCL again, we went through the same exercise for the data structure control structures how to manipulate TCL, and some of the variable substitution, and more the variable substitution with the dollar, and we also had like some functional function substitutions basically that we looked at how to run function within function using the square brackets.

And finally we also looked at the TK essentially and then we looked at how the TK you can write widgets basically and then code all these graphical information what kind of graphical structures are supported by TK, and then we also looked at some of the advanced concepts in TCL, and then finally in the last lecture we also covered some of the uniqueness with the synopsis TCL.

Mainly the, the collections as one of the data types or data objects and how to manipulate the collections, I hope like you remember all the things like they get so some get commands essentially so I think like today we are going to switch gears again and then move to the next topic which is the Python programming, so I will be talking about Python and then also we will be finishing up with this Python programming as the main lecture, so let us begin in earnest with Python programming.

(Refer Slide Time: 03:00)



List of Chapters

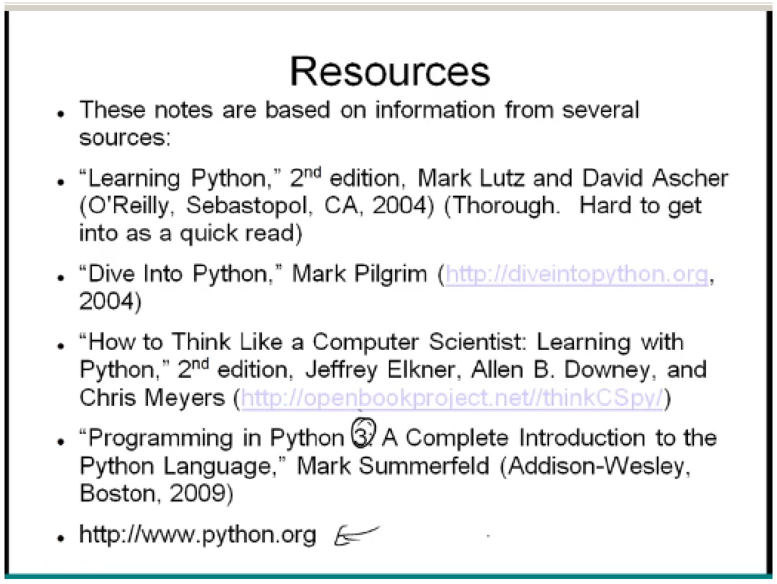
Chapter 1: Very Basic Stuff
Chapter 2: Conditionals
Chapter 3: Functions
Chapter 4: Iteration
Chapter 5: Strings
Chapter 6: Collection Data Types
Chapter 7: Advanced Functions
Chapter 8: Exception Handling
Chapter 9: Python Modules
Chapter 10: Files
Chapter 11: Documentation
Chapter 12: Classes
Chapter 13: CGI Programming

So what we will be talking about we have about 13 chapters that we will be going through, first we look , look at some very basic stuffs as to how to interact with Python how to do some simple commands things like that, then we will go more into details the conditionals, then we will look at the functions the Python functions then we will go through the iteration or the while loops in the for each loops and how to specify those things in Python.

Then we will look at strings the collection data type which is introduced in the typical programming but we will be looking into that, then we will go through some advanced functions , exception handling then we will cover the Python modules, so one thing is like why important we will come to know in a short while, and then we will talk in more details later on then we will look at some of the click on files the documentation classes again this is another e the topic.

Why do we do this and then the CJ programming so these are the main topics that we will be covering in this section of the of our class, so without too much so before we talk about all these things I want make sure that the Python language, is similar to pearls C and Java, but it is only similar there are some definite differences between these languages also, so initially I want to just give you some brief overview of what Python is and then we will go into more details.

(Refer Slide Time: 04:59)

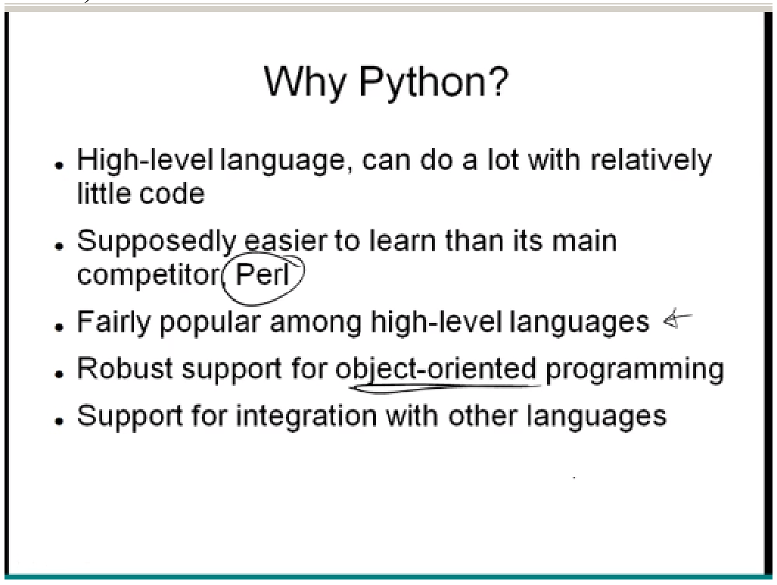

Resources

- These notes are based on information from several sources:
- "Learning Python," 2nd edition, Mark Lutz and David Ascher (O'Reilly, Sebastopol, CA, 2004) (Thorough. Hard to get into as a quick read)
- "Dive Into Python," Mark Pilgrim (http://diveintopython.org, 2004)
- "How to Think Like a Computer Scientist: Learning with Python," 2nd edition, Jeffrey Elkner, Allen B. Downey, and Chris Meyers (http://openbookproject.net//thinkCSpy/)
- "Programming in Python 3: A Complete Introduction to the Python Language," Mark Summerfeld (Addison-Wesley, Boston, 2009)
- http://www.python.org

Before that some of the resources that you can look forward to, so again there is learning Python by Mark Lutz and David Ascher this is one of the books, dive into Python mark pilgrim how to think like a computer scientist, learning with Python is another one, this is by Jeffery Elkner Downey and Chris Meyers, and then the pipe programming in Python 3 is an introduction to Python language again, we will be talking about this number 3 and then there will be also number 2 .

So what is the what is the significance of that we will talk about that, and then finally like I mean this is a kind of a good reference basically, that you can always go back to which is the python.org, this provides you with if you are in doubt you can quickly go to this website and check the answer for your particular problems.

(Refer Slide Time: 06:08)


Why Python?

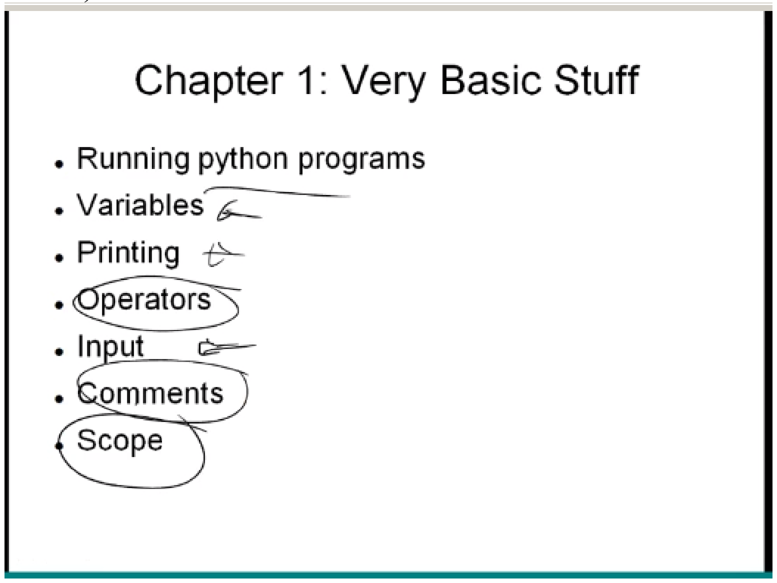
- High-level language, can do a lot with relatively little code
- Supposedly easier to learn than its main competitor Perl
- Fairly popular among high-level languages
- Robust support for object-oriented programming
- Support for integration with other languages

So the first question we ask is now we learned about Pearl learned about TCL, why Python so Python is a high level language, and you can do a lot with relatively little code pearl you can see basically that we can get some person, TCL at you know like I mean it was developed as a quick scripting language, but even in typical programming TCL becomes quite cumbersome when it comes to compose systems.

So why Python it is also supposed to be easier to learn than the main competitor pull so it kind of replaces fill with the more definite concepts explained to you those things, in Python and nowadays is fairly popular among other languages, in fact some of the companies are built on Python for example Google, Google people used by Python quite frequently basically that is the main programming language, and it has a robust support or script, and also it is a it has a robust support for object oriented programming.

So this is the ligament we talked about classes and these things we will talk about that in the coming lectures, and then it also supports integration with other programming language, so basically like if I have Perl and things like that different modules you can easily call from Python language into those, those modules okay.

(Refer Slide Time: 07:59)

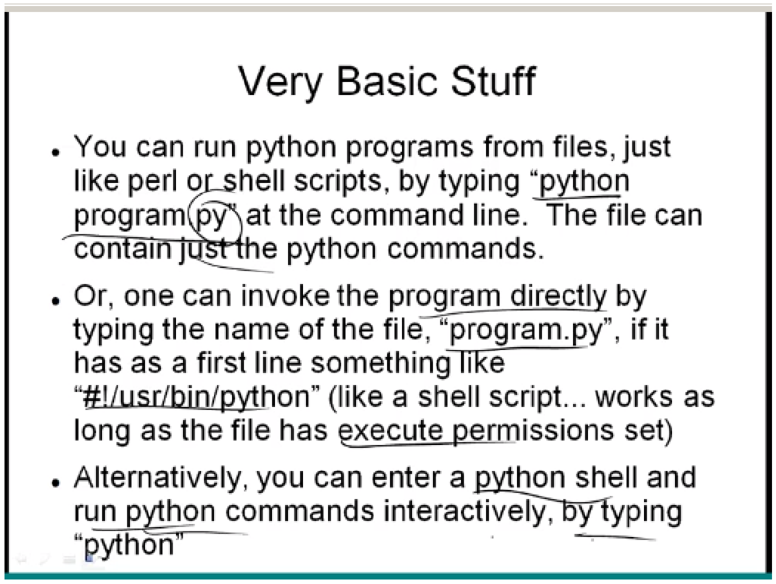


Chapter 1: Very Basic Stuff

- Running python programs
- Variables
- Printing
- Operators
- Input
- Comments
- Scope

So now we will be going into like the very basic stuff the very basic stuff involves running some few Python programs, then we will go into the variables how to print so the printing itself and then some operators, input how do we input objects into a Python program, and how do we call comments and then scope, the scope of variables for how to of the extent, and even like scope of the program itself.

(Refer Slide Time: 08:44)


Very Basic Stuff

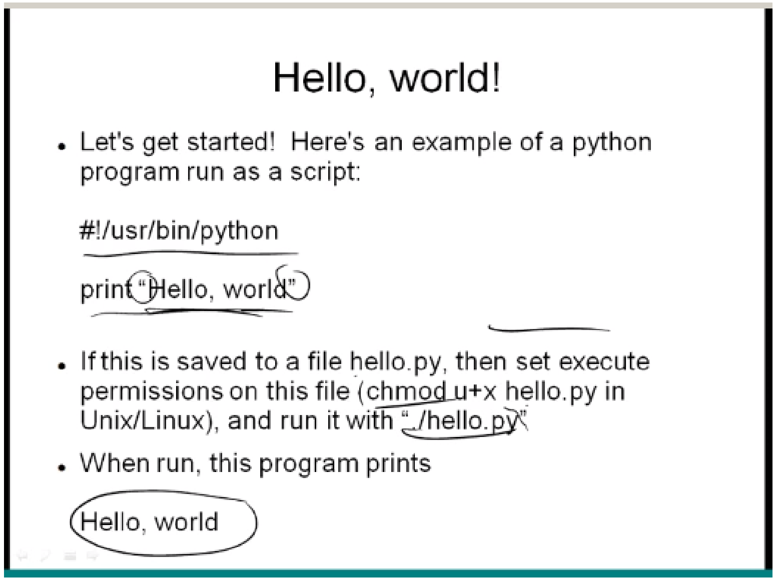
- You can run python programs from files, just like perl or shell scripts, by typing "python program.py" at the command line. The file can contain just the python commands.
- Or, one can invoke the program directly by typing the name of the file, "program.py", if it has as a first line something like "#!/usr/bin/python" (like a shell script... works as long as the file has execute permissions set)
- Alternatively, you can enter a python shell and run python commands interactively, by typing "python"

so we will take one by one, so the Python programs basically you can run Python programs from files just like Perl or shell scripts, by typing Python followed by the program name dot type this py is basically the extension that we will be using, I mean and in fact this is the show give away that is a Python program, the file can just you can it contains just the Python commands citizenship, you can also like invoke the program directly by typing the name of file this program dot file.

If it has the first line the pointer to the Python installation, so here the installation is you could be Python, so if you do like a hash bang user bin python and the first line in your script, and then you followed by the Python commands then you can just simply write that particular file name, and it will execute but we need to set the execution permissions, once you do that it is just execute.

So this is just like a shell script it works as long as the 504 function as I mention, and then finally like I mean you can also enter Python shell, and then the Python commands you can run Python commands interactively by typing Python.

(Refer Slide Time: 10:16)



Hello, world!

- Let's get started! Here's an example of a python program run as a script:

```
#!/usr/bin/python
print "Hello, world"
```

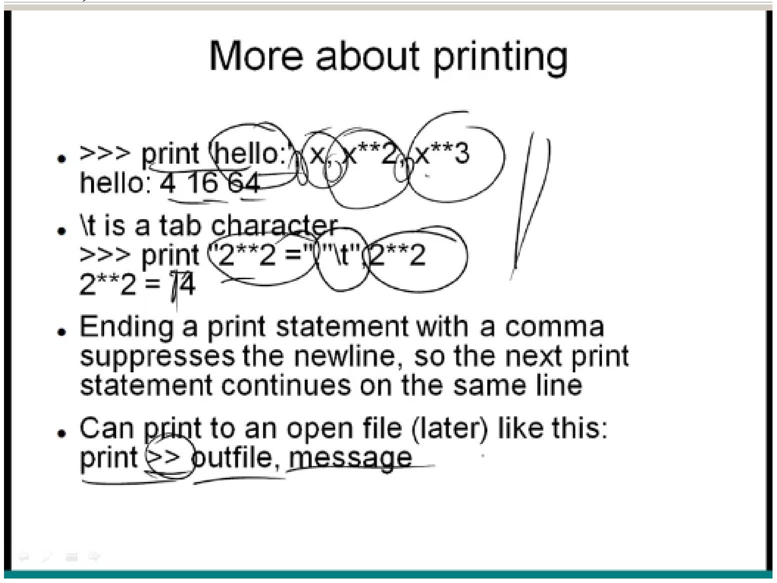
- If this is saved to a file hello.py, then set execute permissions on this file (chmod u+x hello.py in Unix/Linux), and run it with "./hello.py"
- When run, this program prints

```
Hello, world
```

so these are all like valid ways to do it, and now so now let us see how to ruin hello world program, so here is what basically like I mean every programming language we have this hello world script, so here the example is shown here we use the user bin Python, and then just simply say print in ports hello would and if you save this file into a Hello dot py then you can execute the permission on the file basically we will just do a change move you plus X hello py in the UNIX window and then you can just run it as a lot of Py.

When you run this program the output is here basically just means this statement hello both, so this is simple enough I think the you should be like now very familiar it should be confident to write will Python programs.

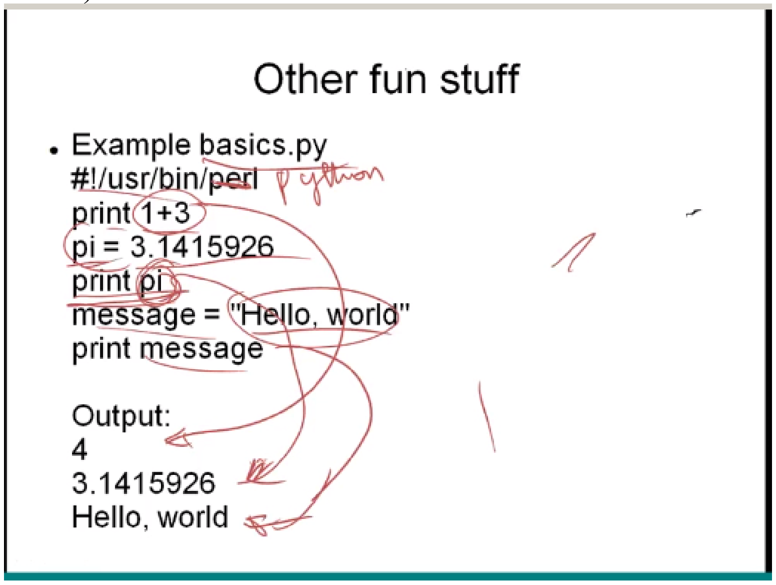
(Refer Slide Time: 11:40)



More about printing

- `>>> print 'hello:', x, x**2, x**3`
  `hello: 4 16 64`
- \t is a tab character
  `>>> print "2**2 =","\t",2**2`
  `2**2 = 4`
- Ending a print statement with a comma suppresses the newline, so the next print statement continues on the same line
- Can print to an open file (later) like this:
  `print >> outfile, message`

So now let us look at some more on the printing so here we say print hello, and then we specify a comma and then X, X**2, X**3, so what this means is essentially it is going to print this string all Hello, followed by whatever the value of X then we X power 2 which is X squared and then within X cubed , so here X is 4 so it printed for 16 and 64 and then whenever we specify backslash T that is a tab character so then it actually like moves it by a tab amount and then prints the next one.
So if you say print 2 times 2 is = and then within port so this is not a value of about the string and then here we just evaluate 2 power 2 arithmetic, so the printing you can see now that you can mix and match things basically you can mix variables with expressions with different types of variables, so this is all like the flexibility that Python office, and if you end a print statement with a comma, that suppresses the new line, so that the next print statement starts at the same.
You can also print through an open file and basically like it is just like this will go out then I mean print and then double arrow say out file and then the message, so the message will be printed.
(Refer Slide Time: 13:37)



Other fun stuff

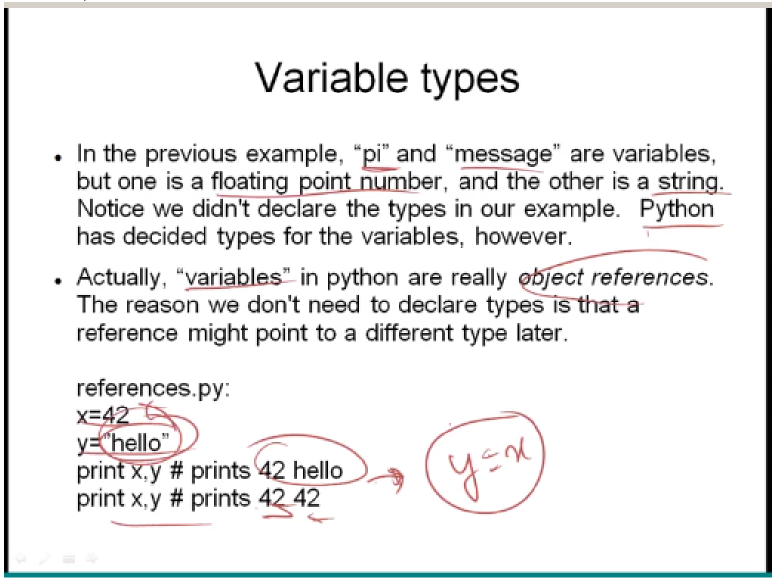
- Example basics.py

```
#!/usr/bin/~~perl~~ python
print 1+3
pi = 3.1415926
print pi
message = "Hello, world"
print message

Output:
4
3.1415926
Hello, world
```

So now let us look at some more fun stuff you so here we have another example the basic of py, so we specify the header actually like this should Python is been Python, then we say print 1 + 3 and we say something like pi is 3.1415926 and you say print pi we can also make a like a message and the message is a string hello world and then we can put print message, so now the output of this particular run is going to be the first one is it is going to evaluate this expression and prints the result which is 4.

For the second one there is a print statement here this print by and then it is only going to replace this five with the actual value, so this is the variable and then it replaces the variable with value, and then finally the last one the print message is again the message will be replaced with its own value, which is the string variable so you look at this basically you have a variable name which you never declare what type of a variable is, and then you also like you can print those things fairly easy.

(Refer Slide Time: 15:43)



Variable types

- In the previous example, "pi" and "message" are variables, but one is a floating point number, and the other is a string. Notice we didn't declare the types in our example. Python has decided types for the variables, however.
- Actually, "variables" in python are really *object references*. The reason we don't need to declare types is that a reference might point to a different type later.

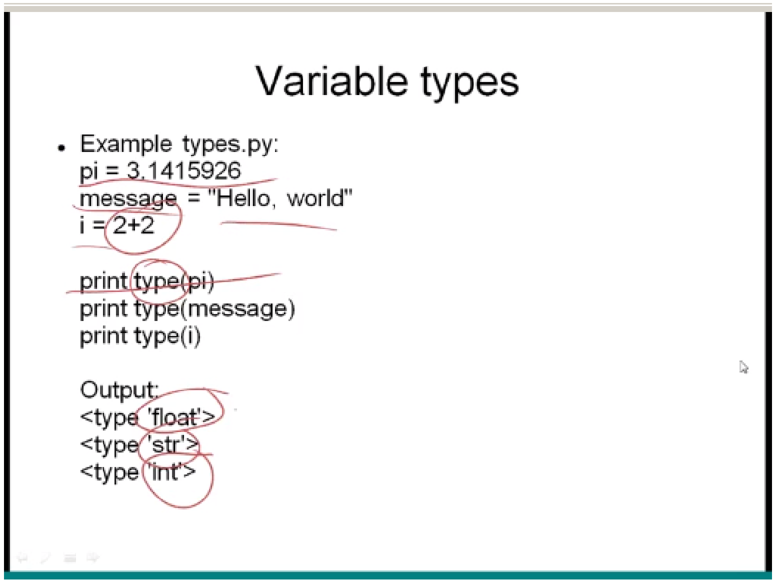
```
references.py:
x=42
y="hello"
print x,y # prints 42 hello
print x,y # prints 42 42
```

So summarizing the previous example as you see like moving the pie and messages message are variables but one is actually a floating point number the other one is a spring, and notice that we never declared this types in our example and Python itself decided what types of these variables, so in Python actually the variables concept of variables at this it is just an object reference this is also like we saw some of these kind of concepts in TCL as a, when we talked about like the associative arrays in fact that they are indexed and value or the index is basically just a place although business is not the preference.

The reason we do not declare the types are because or weapons might pop into different type later on so here is one example X is 42 Y is hello so print X Y then 42 hello bring X comma Y, but you can also change this type into like I mean actually like there is this statement another statement missing with form y = X, and then when you do print X Y now it actually prints 42 and 42 because now it actually like now change the data tag from the string back to this integer.

So you can actually like in Midway you can change the type of the particular variable, so again this is another key concept with Python.

(Refer Slide Time: 17:44)



Variable types

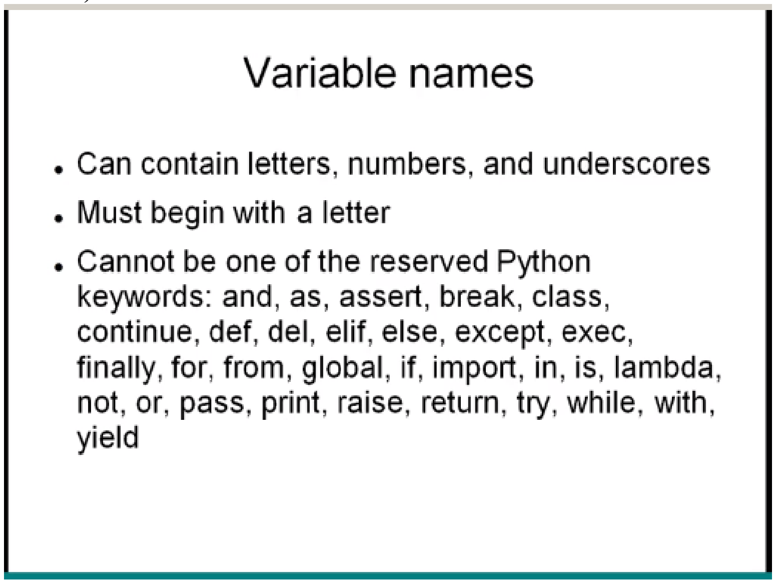
- Example types.py:

```
pi = 3.1415926
message = "Hello, world"
i = 2+2

print type(pi)
print type(message)
print type(i)

Output:
<type 'float'>
<type 'str'>
<type 'int'>
```

So one thing that you can do is you can have so here is there is a function call, we have not introduced the function but let us say like I mean there is a function all the type that determines what kind of type it is, so here this divine pi as 3.1415926 the message is defined as hello world, and then I is an expression two plus two now if you ask Python to print the type of pi type of message and type of I you get this outputs basically the first one is a floating point number the second one string and the third one is an integer.

So it does recognize all the types only thing is it is a intelligent enough to figure out how to apply those steps, and also then to apply what type, so that is a very powerful concept.

(Refer Slide Time: 19:02)

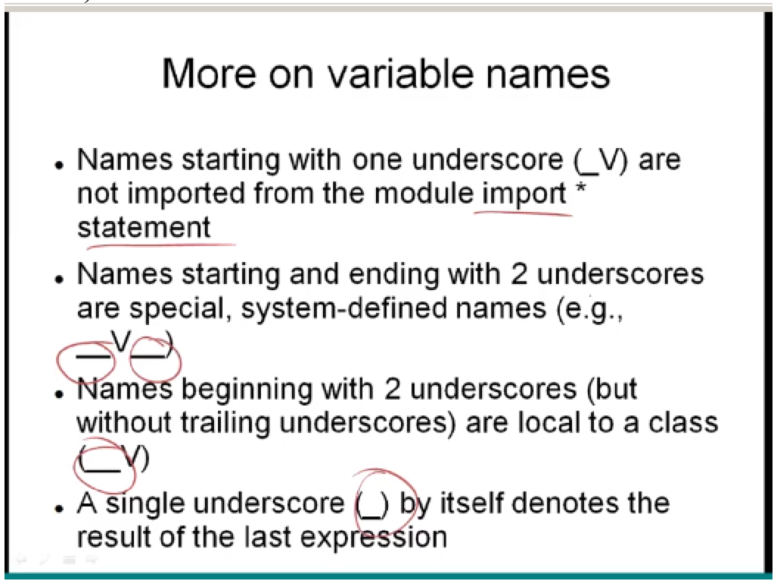


Variable names

- Can contain letters, numbers, and underscores
- Must begin with a letter
- Cannot be one of the reserved Python keywords: and, as, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while, with, yield

So how do we define the variable means, so the variable names can contain letters numbers and underscores, but it has to begin with a letter and that is how Python identifies a variable name,

and it cannot be one of the reserved Python keywords there are some of the examples and, as assert, break, class, continue, def, del, elif, else except, exec, finally, for, from, global, if, import in, if, lambda, not, or, pass, print, raise, return, try, while, bit and yield actually like I mean if you go to any of the websites that I mentioned earlier.
You should be able to find these reserved words then, the thing is basically you cannot use reserved words or a constant or variable or any other identifiable, the identifying name basically our variable name it is used to identify not only just a variable this these rules apply for function a class a module or any other object, and the other thing is a Python also does not allow punctuation characters, such as the add, symbol, dollar or percentage beginning here in times.
Python is a case sensitive programming language so that the uppercase and the lowercase of two different variables to device, so now let us look at someone.
(Refer Slide Time: 21:25)



More on variable names

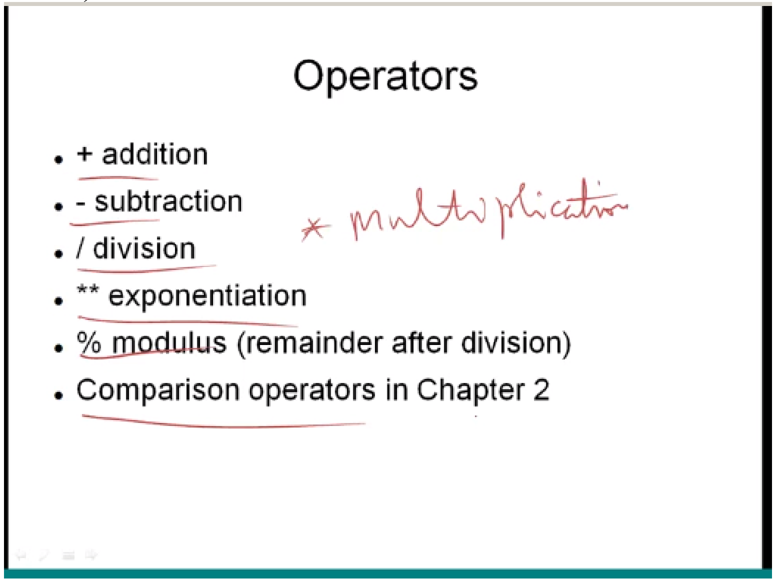
- Names starting with one underscore (_V) are not imported from the module import * statement
- Names starting and ending with 2 underscores are special, system-defined names (e.g., __V__)
- Names beginning with 2 underscores (but without trailing underscores) are local to a class (__V)
- A single underscore (_) by itself denotes the result of the last expression

So any names starting with one number for this being underscore followed by like a letter, they are not imported from the module import statement, we will we will look at this one later on and how to use that, and then names starting and ending with two underscores so exam underscore, underscore V underscore, underscore, they are special we see they are system defined names, the names beginning with two underscores, but without any trialing, trailing underscore they are local to a class.
So when you define a class the variables within the class all of this convention which is for 200 scores followed by the name, a single underscore by itself denotes the result of last expression, so this is again useful when we talk about functions, because one of the things that gets returned

is underscore if you do not specify a specific open statement this is something that we saw in the TCL site basically, so those are the concepts that are still continuing in the in Python.

So the key thing is basically the variables in Python there are nothing but reserved memory locations to store values, this means that you can create a variable and you reserve some space in the member that is all, and based on the data type of variable the interpreter allocates memory and decides what can be stored in the reservoir, so you can store integers decimals or characters into these variables, and icon variables do not have explicitly declared do not have to be explicitly declined to reserve them in space so this is what we saw in the earlier ones .

(Refer Slide Time: 24:18)



Operators

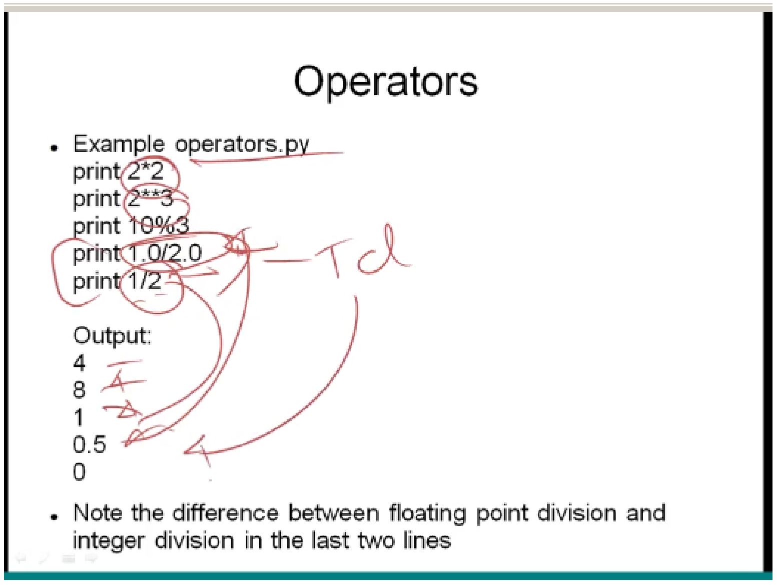
- + addition
- - subtraction
- / division
- ** exponentiation
- % modulus (remainder after division)
- Comparison operators in Chapter 2

* multiplication

So now let us look at some other operators form so the standard operators basically that we all go form plus sign for addition, minus for subtraction, division, exponentiation we also have star go for multiplication, and then model is basically which is a remainder after division and then we will talk about the comparison operators in subjective, one thing that I kind of talked about earlier basically is this concept of simplification basically this is a various this is kind of a it is not a purely a scripting language.

But kind of high level language which also enables you to program in lot of shortcuts so one particular shortcut probably like we will talk about this is this multiple assignment, you can do something like A = B = C = 1 this is like a basic evidence assigned to one you can also say A, B, C = 1 2 3 so everything is assign once so A assign 1B assign 2 and C assign 3, so we will talk about some of these things also I mean slides.

(Refer Slide Time: 26:00)



Operators

- Example operators.py

```
print 2*2
print 2**3
print 10%3
print 1.0/2.0
print 1/2
```

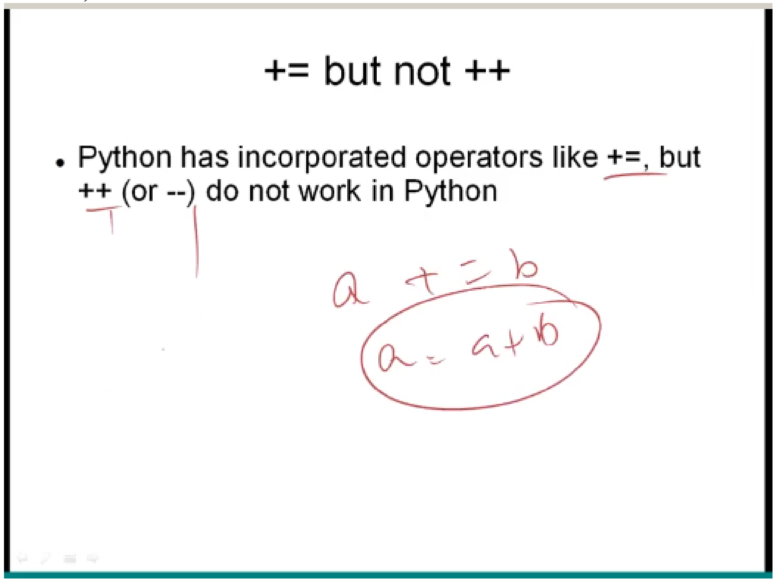
Output:

```
4
8
1
0.5
0
```

- Note the difference between floating point division and integer division in the last two lines

So here is another example someone this is known as the operators Python, so you know what 2 times 2 is so is 4 now if you do a 2**3 it is explanation the result is 8 and if you do a 10 % 3 the answer is 1, now if you print 1.0 / 2.0, this is something that we find in TCL this is a floating point so the result is 0.5 but if you do 1/2 that is still TCL a integers so the result is 0, so this is exactly the same that we saw in TCL as a so in that sense basically like TCL.

(Refer Slide Time: 27:05)



+= but not ++

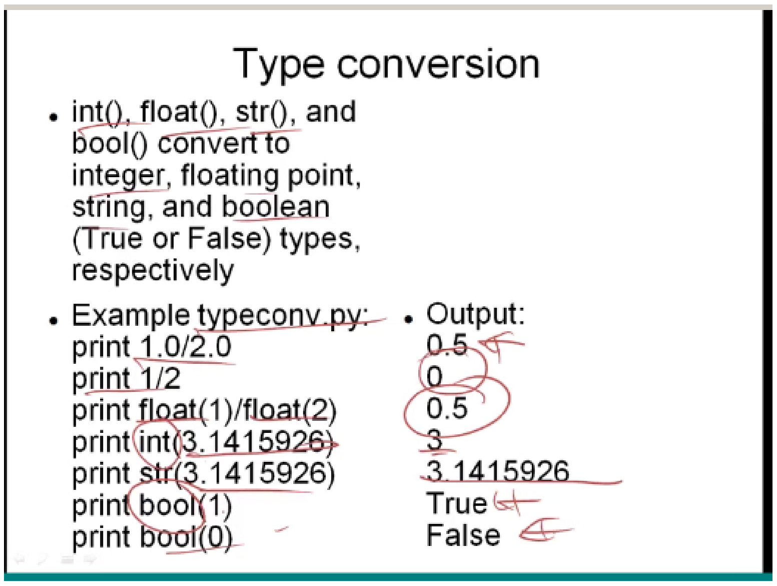
- Python has incorporated operators like +=, but ++ (or --) do not work in Python

$a += b$

$\boxed{a = a + b}$

Now Python is actually incorporated operators like + = which is like A+ = B same as A = A + so it is like the self addition but the ++ increment or decrement operations they do not work in Python.

(Refer Slide Time: 27:37)



Type conversion

- int(), float(), str(), and bool() convert to integer, floating point, string, and boolean (True or False) types, respectively
- Example typeconv.py:

```
print 1.0/2.0
print 1/2
print float(1)/float(2)
print int(3.1415926)
print str(3.1415926)
print bool(1)
print bool(0)
```

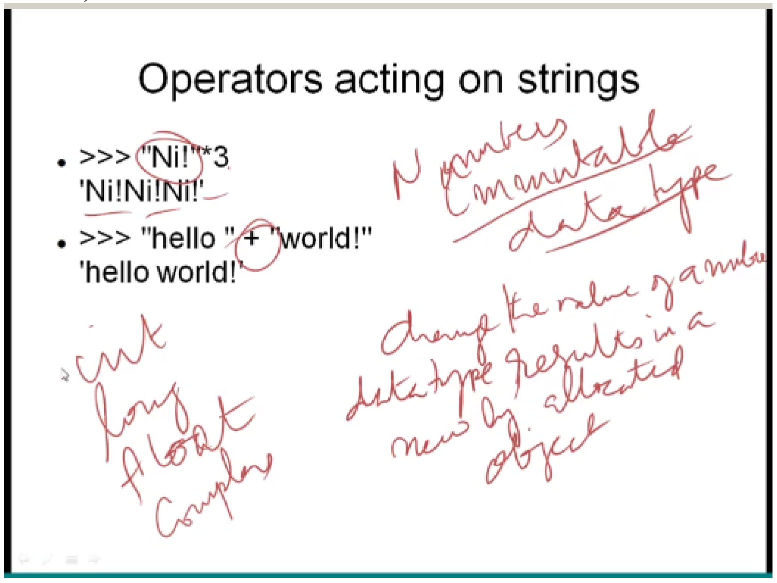
- Output:

```
0.5
0
0.5
3
3.1415926
True
False
```

So now let us look at some of the type conversions essentially, any int, float, strain, booling or converted to integer floating-point string or Boolean the Boolean is essentially will have just true or false respectively, so here is a type conversion routine, if you do a print 1.0 by 2.0 the output is five, five print 1/2 and integers so output is 0 , you can also mean specifically mention it is float 1 / floor 2 2 and the result is point 5.

And now you can convert this floating point into integers, would not still be operating upon an integer and then the result is 6 3 and if you do a string the result is exactly the same because in string actually they do not have you mean this book just the position now the bool function basically of one is true and then both function of 0 is false.

(Refer Slide Time: 29:06)



Operators acting on strings

- >>> "Ni!"*3
  'Ni!Ni!Ni!'
- >>> "hello " + "world!"
  'hello world!'

N numbers
Immutable data type

change the value of a number datatype results in a newly allocated object

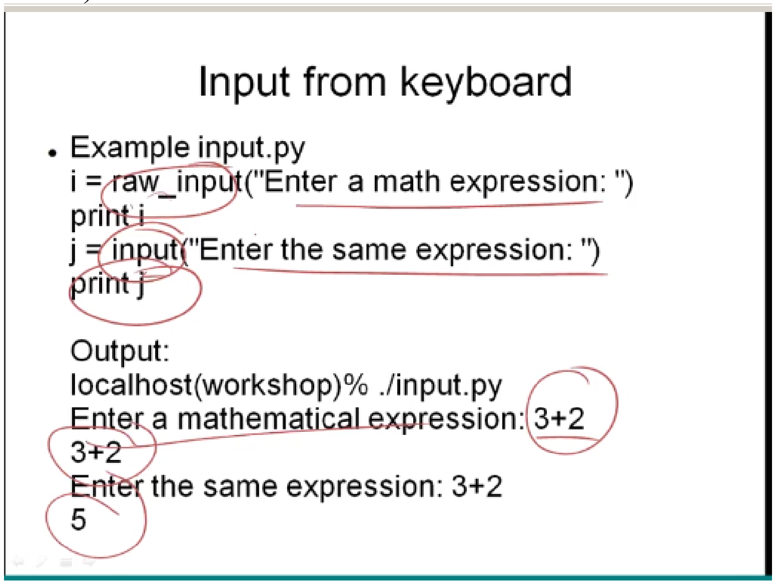
int
long
float
complex

Now some of the things that we saw in the article and that is still applicable in Perl for example this string multiply so if you have a string operator, times 3 is B both instantly and then you can also use a plus operator or an addition operator on string, and it just prints out, so before we go into the next one I want to talk about other things one is the numbers in Python the, the data type is immutable this is a new term in business.

So immutable data type, what this means is the changing the value of that number data type results in a newly allocated object, so you could change the value of a number data type those in a newly allocated, so numbers are immutable get back, a number of death is created when you are saying value to any of the object, and the Python supports four different numerical types, we saw one of them in the previous one int and it also supports long they are called the long integers essentially they usually it is represented in octal or hexadecimal.

Then float which we saw with example and then finally complex and then the complex is basically for complex numbers.

(Refer Slide Time: 32:23)



Input from keyboard

- Example input.py

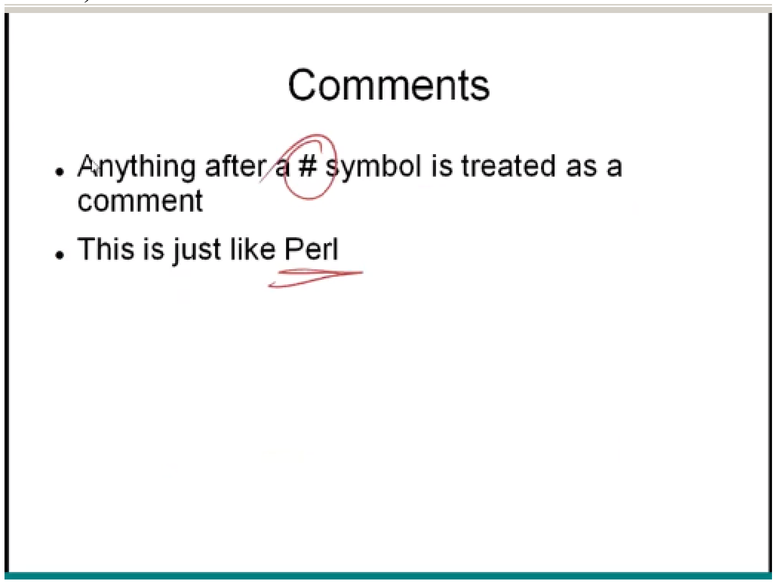
```
i = raw_input("Enter a math expression: ")
print i
j = input("Enter the same expression: ")
print j
```

```
Output:
localhost(workshop)% ./input.py
Enter a mathematical expression: 3+2
3+2
Enter the same expression: 3+2
5
```

Now let us look at how do we gather input from the keyboard so for this we use this particular function raw input or we can also use this input so let us see like how we can make this opposed so I = raw input enter the Math expression, so and then we print I so it says basically leg into the math expression and then we really 3+2 now the I value is the same P + 2 does not do anything, basically the public into do I.

Now we say like J = input into the same expression now if you entered 3+2 it is evaluated and then the results shown as value of T, so we have like two rate of inputting value or, or data from keyboard one is the raw input and then the other one is input, so the raw input just obvious

whatever you type exactly as into the take it takes it into the program, whereas if you do a just an input it evaluates the expression first and then force the result into the variable.
(Refer Slide Time: 34:02)



Comments

- Anything after a # symbol is treated as a comment
- This is just like Perl

Now how do we comment use a comment in Python it is very simple, same similar basically the hash sign is views as a comment, and anything in a line after this hash symbol is trivial and second so you can put anywhere the comment anywhere, and it will just be treated as so coming from that point onwards in that line it is just like the Perl ,so even in Perl we do exactly okay so.
So far what we have done is just introduced the Python so we started with actually what why Python, this I think you had understood that concept basically, we talked about basic stuff which is how do we invoke a Python, first of all we talked about the high level language how, how we can write code or relatively less code, and how we can write the code that are easier than for, and also like I mean how it is popular basically Python is being used in several companies as the main programming environment.
It also supports object oriented Ness for which we will talk about in towards the end of Python, but I want to introduce to the object oriented programming concepts, and then finally liked it also allows integration with other languages, now we also like talked about some of the things we wanted to see how to run Python programs, or our variables how do we print those things and then some of the operators and it is so, input to them how do we input into the program into the Python and then the comments.
We have not talked much about the scope you will see like within talk now, so for invoking the program the Python program essentially in this type Python followed by Python program, or the other method is basically put the back to the Python and you can invoke it from there, or the third

method is basically linking this local Python shell and wonderful compliments interactively by just typing Python follow.
So simple example hello world with troubles with just a print and any, some of the other ones basically want to pinch like the value of X squared and X cube it is very easy below and then all this and it just prints exactly see, you can also print like the strings this is the whole thing is a string followed by a tab and then the actual expression, so the extraction expression will be evaluated in the values shown.
And we can also print the result or some any kind of message into a file by just using the double arrow and then the up file, so we saw some of the variable types essentially ligament can be floating poking the string or of an integer, and they also mentioned that and have made multiple arguments, and then you variable type can actually change momentum here initially it was a string type and then integer type by just doing this.
And then also mention that these variables are immutable, so and you know the definition, and then the type basically type is a function that displays the type of variable, so even the street type I type message type I then prints out each for each of them what is the best time for me to use, then we saw the rules can rules for the variable names essentially we are going to contain letters numbers can contain letters numbers of course but it needs to begin with a letter and it cannot be any one of the reserved words in Python.
So those are the things that one thing that is basically like is another key element in Python that we can talk about is the indentation, so the, the lines and indentation are very crucial in Python, so if you say like I mean true and print then we fit no then another point so it knows that different layers hierarchy through this the indentation, so it is very, very important to date you are lying so that it does not doing that.
We also saw like to the some more rules for the variable names, so single underscore essentially like I mean this is this is to show that they are not imported from the model import statement and then the, the name is starting with double underscore that is a special order system defining and if the names, begin with the double underscore followed by the variable name, without the trailing underscores then it is a local to a class so, so within the class the variables that are specified are systems.
And then finally a single underscore itself or the result of the last expression so they mentioned you can use it for functions to return values, now we look at operators some of the operator is going to be plus, minus, division, exponentiation, modulus then more so the comparison operators they leave usable, so here is just an example of all the operators essentially when you

can go through at your leisure, you can see like how they translated and what is up or each of these expressions.
The key differences here basically how do we do the putting point division versus a normal division and if you use this thing basically again you want to make sure that your answer is correct and then it has this + = operator which is combining the assignment and that we actually though already does not have an increment or decrement type of operative or mission, on the string data type but now the Python can accept single coat or a double coat or even a triple coat and follow coat on a quadruple code to denote strings string literals.
As long as the same type of code starts and ends the string, the triple coat is usually used for Strings that go across multiple lines, so if you want to stand across the string because one length then the new triple coats, and a single code essentially it also legal syntax, and then we looked at how do we do an operations on strings like this the times mean becomes an 3 or hello plus world becomes hello.
And if you want input from a keyboard the raw input basically and input are the two functions available, so the raw input actually gets whatever is there as is and then as the math expression will try to evaluate an expression and that is what gives the input value, if you use input then this mathematical expression is evaluated is raw input then it is not valid, let me look at the comments.
You know that anything after the hash symbol is a treated as a comment it is just like a Perl, so I think we will actually anything like covered most of the things that we said we will be covering except the scope of the variables, we will talk about the scope in the, the next lecture of the extent to which the variable names can be and be available so that is something that we will talk about it in the next lecture okay thank you very much.