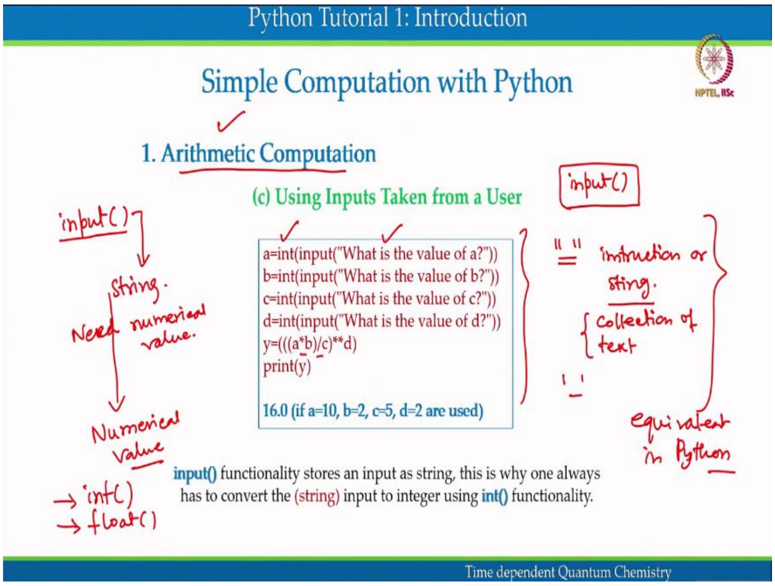
Time Dependent Quantum Chemistry
Professor. Atanu Bhattacharya
Department of Inorganic and Physical Chemistry
Indian Institute of Science, Bengaluru
Lecture No. 06
Simple Computation with Python Programming

Welcome back to Python tutorial 1 of this course, Time Dependent Quantum Chemistry. In this tutorial, we are learning Python for the first time and so far, we have shown how to perform simple arithmetic computation.

(Refer Slide Time: 00:42)



Python Tutorial 1: Introduction

Simple Computation with Python

1. Arithmetic Computation

(c) Using Inputs Taken from a User

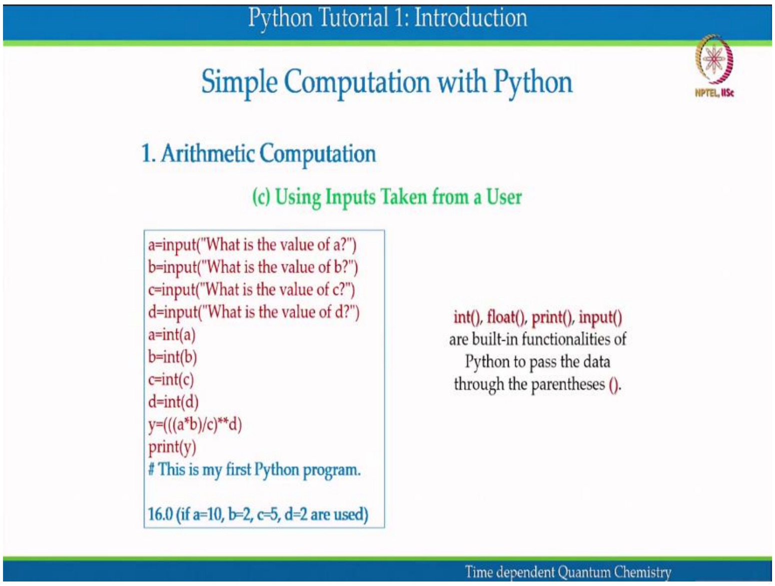
```
a=int(input("What is the value of a?"))
b=int(input("What is the value of b?"))
c=int(input("What is the value of c?"))
d=int(input("What is the value of d?"))
y=(((a*b)/c)**d)
print(y)

16.0 (if a=10, b=2, c=5, d=2 are used)
```

input() functionality stores an input as string, this is why one always has to convert the (string) input to integer using int() functionality.

Time dependent Quantum Chemistry

We have seen that if I take the input from an user, I have to convert it to a numerical value. And that is, there are functionality built in functionality in Python, which you can use either int functionality or float functionality to convert a string to a numerical value.

(Refer Slide Time: 01:05)



Python Tutorial 1: Introduction

Simple Computation with Python

1. Arithmetic Computation

(c) Using Inputs Taken from a User

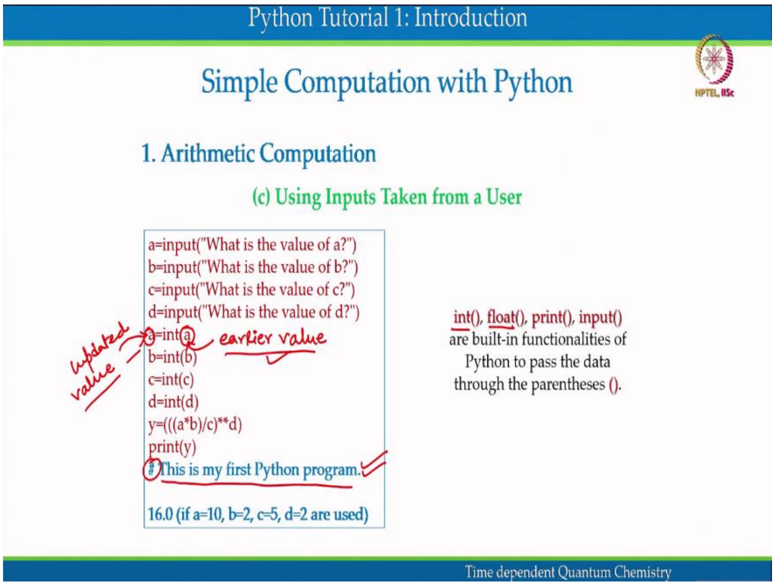
```
a=input("What is the value of a?")
b=input("What is the value of b?")
c=input("What is the value of c?")
d=input("What is the value of d?")
a=int(a)
b=int(b)
c=int(c)
d=int(d)
y=(((a*b)/c)**d)
print(y)
# This is my first Python program.

16.0 (if a=10, b=2, c=5, d=2 are used)
```

int(), float(), print(), input() are built-in functionalities of Python to pass the data through the parentheses ().

Time dependent Quantum Chemistry

Now, this can alternatively be done in a little different way different lines, and that I will show right now in the in my laptop.

(Refer Slide Time: 01:10)



Python Tutorial 1: Introduction

Simple Computation with Python

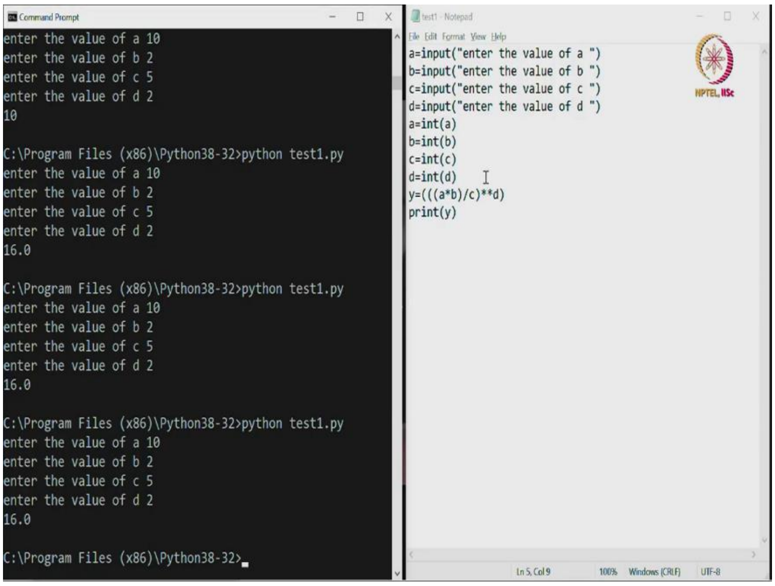
1. Arithmetic Computation

(c) Using Inputs Taken from a User

```
a=input("What is the value of a?")
b=input("What is the value of b?")
c=input("What is the value of c?")
d=input("What is the value of d?")
a=int(a)
b=int(b)
c=int(c)
d=int(d)
y=(((a*b)/c)**d)
print(y)
# This is my first Python program.

16.0 (if a=10, b=2, c=5, d=2 are used)
```

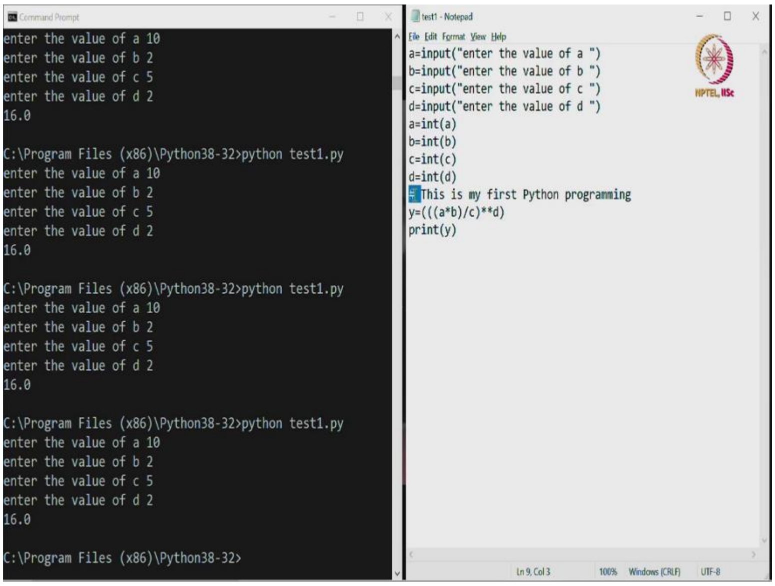
int(), float(), print(), input() are built-in functionalities of Python to pass the data through the parentheses ().

Time dependent Quantum Chemistry



```
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
10

C:\Program Files (x86)\Python38-32>python test1.py
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>python test1.py
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>python test1.py
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>
```

```
a=input("enter the value of a ")
b=input("enter the value of b ")
c=input("enter the value of c ")
d=input("enter the value of d ")
a=int(a)
b=int(b)
c=int(c)
d=int(d)
y=(((a*b)/c)**d)
print(y)
```



```
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>python test1.py
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>python test1.py
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>python test1.py
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>
```

```
a=input("enter the value of a ")
b=input("enter the value of b ")
c=input("enter the value of c ")
d=input("enter the value of d ")
a=int(a)
b=int(b)
c=int(c)
d=int(d)
#This is my first Python programming
y=(((a*b)/c)**d)
print(y)
```

Instead of placing them in the same line, I can always update the value. So, initially, a, b, c, d values will be taken from user, and then we will update that value with the desired format. So, what we will do right now, a equals again, I will write a equals int a, b equals int b, c equals int c, d equals int d. So, if we do that, interesting thing is that, we will go back to the slide.

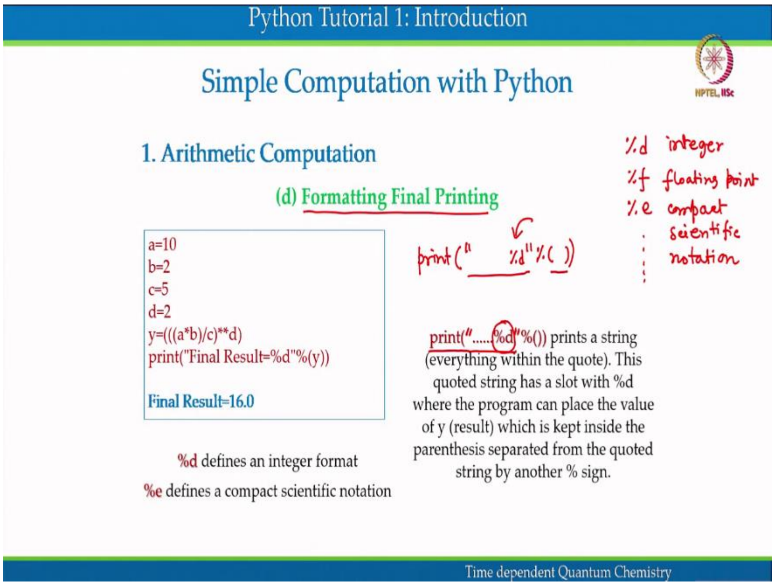
This a is the earlier value and this kind of construct is very useful in computer programming, and this a is the updated value. So, this a is going to be string, and this a is going to be now converted integer, we are converting to integer, we can convert it to float also and then we can perform the task.

So, we will save it in the laptop will perform it, it is asking me the same values, I will be using the same value and I get back the result. So, this is something which you should remember that anytime and, in the slide, anytime one can update a particular variable without changing its name, on the right-hand side, it is giving the earlier value and on the left-hand side it is the updated value and this construct will be used very frequently.

Another thing which we have shown here, something is written a text is written with a hash sign. And this is something which you should remember in Python programming, everything whatever we write after hash will not be executed is not part of the programme, it is just for instruction. So, I will show in the laptop with a hash, if I write down fthis is my first Python programming.

Anything I can write it is the execution is not hampered because of that text is still executing, So, it will ignore when it will ignore that line when it is executing the Python programme. So, anything I can write, after the hash, it is useful, whenever we are writing some instruction for the people who will be using the programme for even me who is writing the programme, for it will remind me what I what instruction I have in the following lines. So, good instruction line would be very useful for any programme to understand without any difficulty.

(Refer Slide Time: 04:59)



Python Tutorial 1: Introduction

Simple Computation with Python

1. Arithmetic Computation

(d) Formatting Final Printing

```
a=10
b=2
c=5
d=2
y=(((a*b)/c)**d)
print("Final Result=%d"%(y))

Final Result=16.0
```

%d defines an integer format

%e defines a compact scientific notation

print(" ___ %d"%( ))

print("......%d"%()) prints a string (everything within the quote). This quoted string has a slot with %d where the program can place the value of y (result) which is kept inside the parenthesis separated from the quoted string by another % sign.

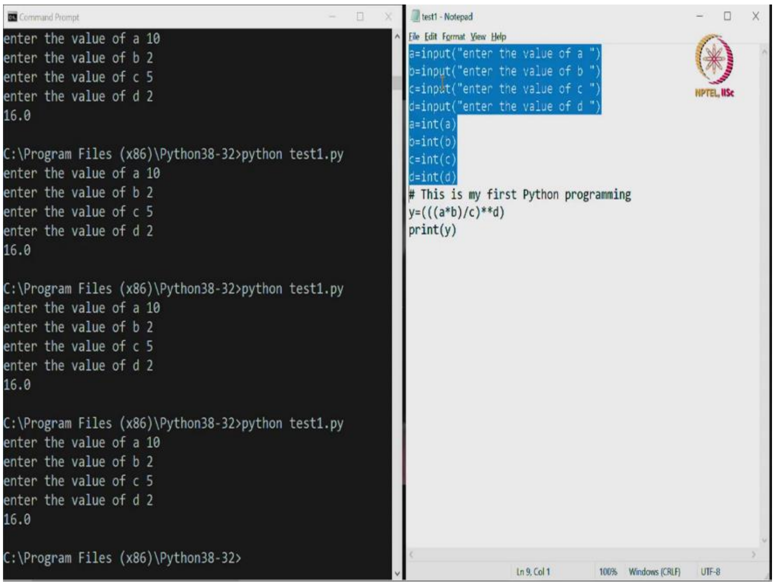
%d integer

%f floating point

%e compact scientific notation

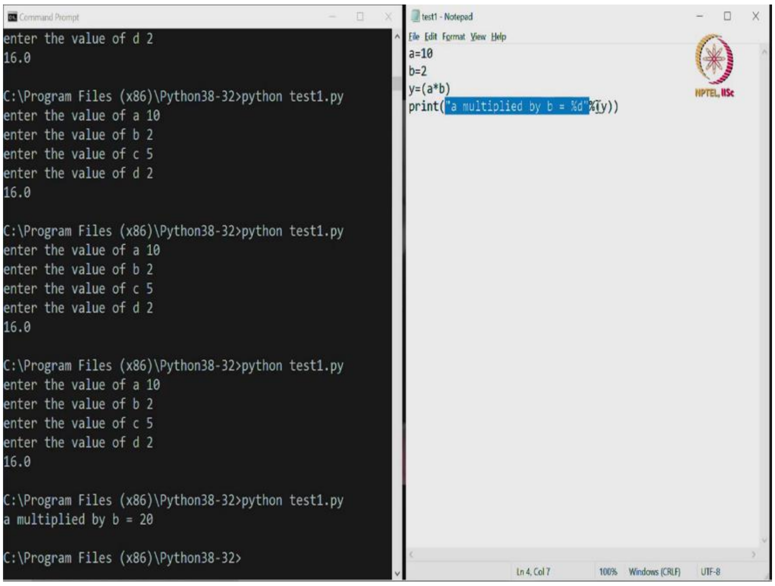
⋮

Time dependent Quantum Chemistry



```
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>python test1.py
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>python test1.py
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>python test1.py
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>
```

```
a=input("enter the value of a ")
b=input("enter the value of b ")
c=input("enter the value of c ")
d=input("enter the value of d ")
a=int(a)
b=int(b)
c=int(c)
d=int(d)
# This is my first Python programming
y=(((a*b)/c)**d)
print(y)
```



```
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>python test1.py
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>python test1.py
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>python test1.py
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>python test1.py
a multiplied by b = 20

C:\Program Files (x86)\Python38-32>
```

```
a=10
b=2
y=(a*b)
print("a multiplied by b = %d"%(y))
```

We will move on and we will check another details of the computation, it is not related to the computation exactly it is related to the final printing format. How we are printing the format? How we are printing the final result? So, formatting the final result is a good idea and we will use this construct, construct is following, print within bracket within quotation we have written a string with a slot given as percent d.

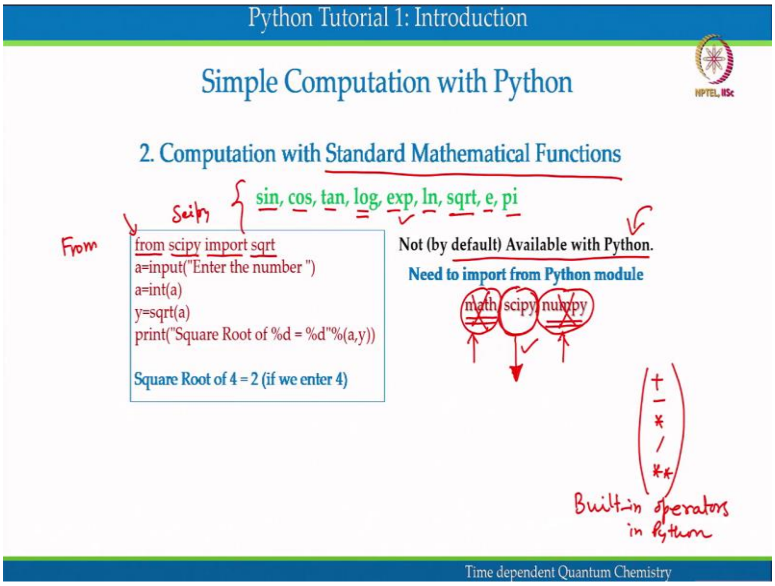
Percent d will be taking the integer value percent f will be taking the floating point value percent e will be taking compact scientific notation. This is floating point and this is for integer. There are many other options which can be navigated in Python dot org, that website I have mentioned in the beginning of this of this tutorial, and one can check what options are available.

So, the basic idea is that within print I will have a quoted string with a slot somewhere indicating certain integer or floating-point value and that value will be inserted by the value given in this bracket and this string and the value, they will be separated by another percent sign. So, that is the formatted output, I will show how to do that in laptop.

So, we do not need input right now, for this demonstration, we will just remove it, we will just write down a equals let us say 10 and b equals 2 and then we will use a multiplied by b this is enough for this demonstration and then instead of print we will give double quotes single quote is also working, will work for Python. So, I will write down that a multiplied by b equals percent d then another percent sign and within bracket this y value.

So, it suggests that while printing the entire string given within this double quote, the entire string will be printed while it is printing, it will fill up the slot percent d by the value of y that is the meaning of this format. So, if I run this programme, it is giving me a multiplied by b equals 20, 20 is the value of y which is showing up right now. This slot has been filled by the value of y and these two sections this quoted string and the value of y should be separated by one percent sign. We have pretty much understood the simple arithmetic computation.

(Refer Slide Time: 09:37)



Python Tutorial 1: Introduction

Simple Computation with Python

2. Computation with Standard Mathematical Functions

Scipy { sin, cos, tan, log, exp, ln, sqrt, e, pi

From

```
from scipy import sqrt
a=input("Enter the number ")
a=int(a)
y=sqrt(a)
print("Square Root of %d = %d"%(a,y))

Square Root of 4 = 2 (if we enter 4)
```

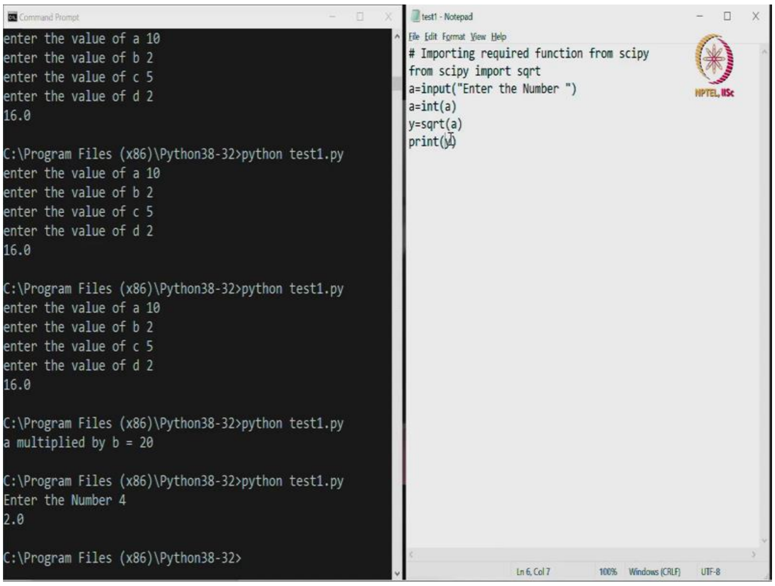
Not (by default) Available with Python.

Need to import from Python module

~~math~~ scipy ~~numpy~~

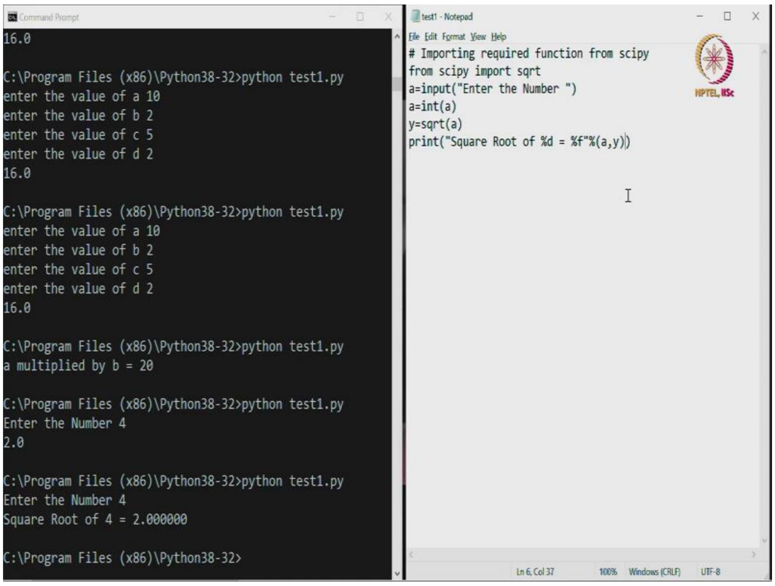
+ − * / ** Built-in operators in Python

Time dependent Quantum Chemistry



```
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>python test1.py
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>python test1.py
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>python test1.py
a multiplied by b = 20

C:\Program Files (x86)\Python38-32>python test1.py
Enter the Number 4
2.0

C:\Program Files (x86)\Python38-32>
```

```
# Importing required function from scipy
from scipy import sqrt
a=input("Enter the Number ")
a=int(a)
y=sqrt(a)
print(y)
```



```
16.0

C:\Program Files (x86)\Python38-32>python test1.py
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>python test1.py
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>python test1.py
a multiplied by b = 20

C:\Program Files (x86)\Python38-32>python test1.py
Enter the Number 4
2.0

C:\Program Files (x86)\Python38-32>python test1.py
Enter the Number 4
Square Root of 4 = 2.000000

C:\Program Files (x86)\Python38-32>
```

```
# Importing required function from scipy
from scipy import sqrt
a=input("Enter the Number ")
a=int(a)
y=sqrt(a)
print("Square Root of %d = %f"%(a,y))
```

We will move on to the computation with standard mathematical functions. Now, here we have to remember something plus, addition, subtraction, multiplication, division, exponentiation. These are all built in functionality or operators in Python. But if I have an mathematical expression with any of the mathematical functions such as sine, cos, tan, log, natural log, 10 base log, then exponential, then square root, then e the value of e or pi any of these mathematical functions if I want to use in Python, they are not directly available with Python.

Depending on the convenience, what scientific community has done, separate library or module has been prepared in Python and in order to access those features in those modules, we have to import the module first and the construct which can be used to import that module is following. But before we import, another thing, we will remember that there are multiple modules available in Python.

For many purpose, let us say there are mathematical functions which can be imported, that is called math module of Python, there are numerical modules, which is numpy, that module can also be imported. And also for scientific computing, this scipy module, all these modules are available and each module has its own benefit. But in this course, very frequently, we will be using this scipy module.

So, although if I only considered the standard mathematical functions, which are available currently, sin, cos, tan, trigonometric or exponential all these functions are available with all three modules in all three modules, because for a long term, scientific Python scipy will be used very frequently in this course, we will only use scipy to import the necessary mathematical functions.

So, we will not use math module or numpy module in this in this course, we will use only scipy module. Scipy documentation has been given already in the website which I have mentioned in the beginning of this course. So, you can take a look at the syntax, functionalities, features, numerical recipes, which are available with scipy. So, what is the construct to import some mathematical function from scipy, this is the construct, it is very simple construct from scipy import sqrt, square root this mathematical function you import.

So, we will now demonstrate in the laptop from scipy import sqrt, this mathematical function import I can write down here for my instruction importing required function from scipy this line starts with hash so, it will not be executed while running the Python programme; a equals I will put input, enter the number, it should be within quote, then I have to convert it because it is a numerical computation we will be doing this is coming as string right now a is a string.

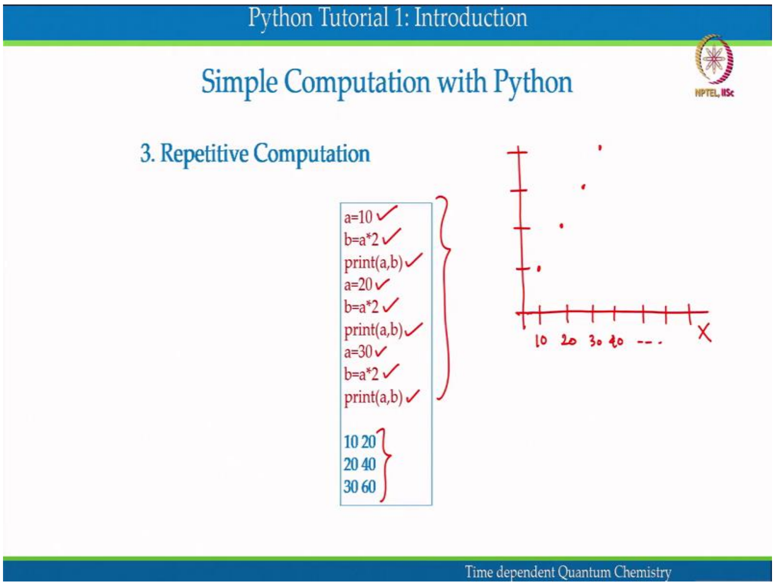
So, we have to move it to convert it to some integer value or float point value. And then I will define y as sqrt of a and then I will print y. I saved it and now I am going to run it. It is asking me to enter the number and the square root of 4 is going to be 2. So, this giving me the result, I will format my print a little bit, square root of percent d I will write, I will be the slot will be filled by a and then equals percent, I will write f.

So, what it suggests that I have now to string I have a long string, which will be displayed, which will be printed and, in this string, I have two slots percent d, and percent f these two slots I have and in the same order, I have given the values of the slots, the first value percent d will be filled by a and the second slot percentage f will be filled by y and that is the way we can maintain the sequence.

So, let us check enter the value 4 is giving square root of 4 the 4 value, which is a value percent d has been filled by this a and equals next slot percent f has been filled by y that is 2.000, 2.000 is giving like this way because I have selected the print format as for percent f and that is why it is giving me the this this value. So, this is the way one can import different mathematical function from scipy.

And remember that anywhere by mistake, if you make it capital like From or Scipy, it will give you an error because it will distinguish the capital letter and small letter the Python will distinguish that. What are the different mathematical functions available in scipy as in a general that can be also navigated in that scipy module.

(Refer Slide Time: 17:49)



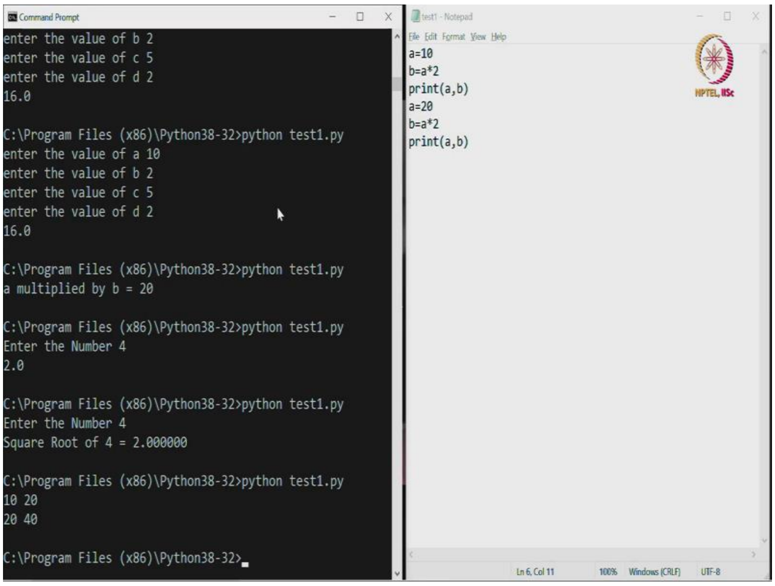
Python Tutorial 1: Introduction

Simple Computation with Python

3. Repetitive Computation

```
a=10 ✓
b=a*2 ✓
print(a,b) ✓
a=20 ✓
b=a*2 ✓
print(a,b) ✓
a=30 ✓
b=a*2 ✓
print(a,b) ✓

10 20
20 40
30 60
```

Time dependent Quantum Chemistry



```
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>python test1.py
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>python test1.py
a multiplied by b = 20

C:\Program Files (x86)\Python38-32>python test1.py
Enter the Number 4
2.0

C:\Program Files (x86)\Python38-32>python test1.py
Enter the Number 4
Square Root of 4 = 2.000000

C:\Program Files (x86)\Python38-32>python test1.py
10 20
20 40

C:\Program Files (x86)\Python38-32>
```

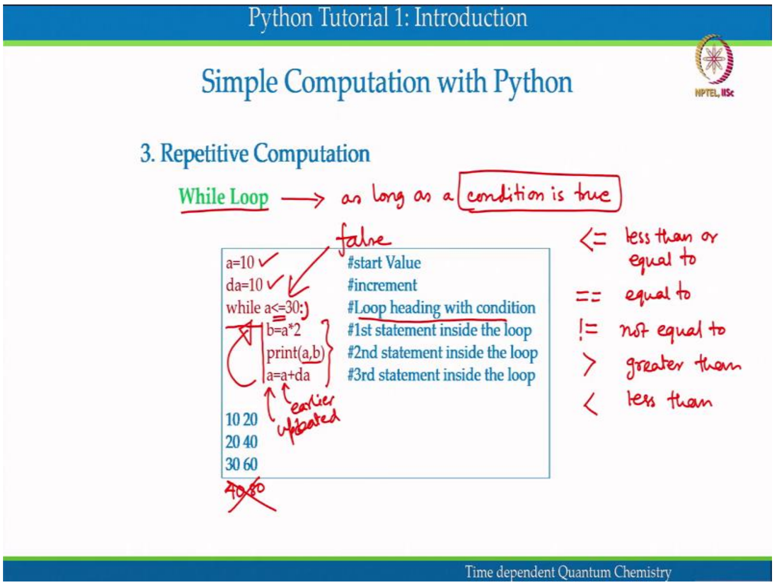
```
a=10
b=a*2
print(a,b)
a=20
b=a*2
print(a,b)
```

So, we will move on to the computation little more complicated as compared to previous ones. Often in different computation, we required to repeat the similar kind of computation several times. So, for example, here the example is given I want to I have a value, then b value is just square of that, and I have to print it again a value will be I will change the value to 20 multiplied by 2 and print a, b again I will have a value of 30, then multiplied by 2 and print a, b and I get this print.

Let us check it if I do it several times a equals 10, b equals a multiplied by 2, print within bracket a comma b, then I will repeat the sequence with a different value of a and if we do it I get 10, 20, 20, 40 prints I have and so on this is the way one can repeat and this kind of repetition is required particularly for the graph plotting many times what we do in the x axis we take a sequence with well specified deviation our difference.

So, we can take 10, 20 then 30, 40 like this way and then we convert the values of y accordingly and we get some values like this. So, we get one point here, another point probably is here and the point is probably here and the point is over here, like this way, we repeat that, and this kind of reputation in computation is required. In many occasions, this is just one example for graph plotting, we use that.

(Refer Slide Time: 20:28)



Python Tutorial 1: Introduction

Simple Computation with Python

3. Repetitive Computation

While Loop → as long as a condition is true

```
a=10                #start Value
da=10               #increment
while a<=30:        #Loop heading with condition
    b=a*2           #1st statement inside the loop
    print(a,b)      #2nd statement inside the loop
    a=a+da          #3rd statement inside the loop

10 20
20 40
30 60
```

~~40 80~~

<= less than or equal to

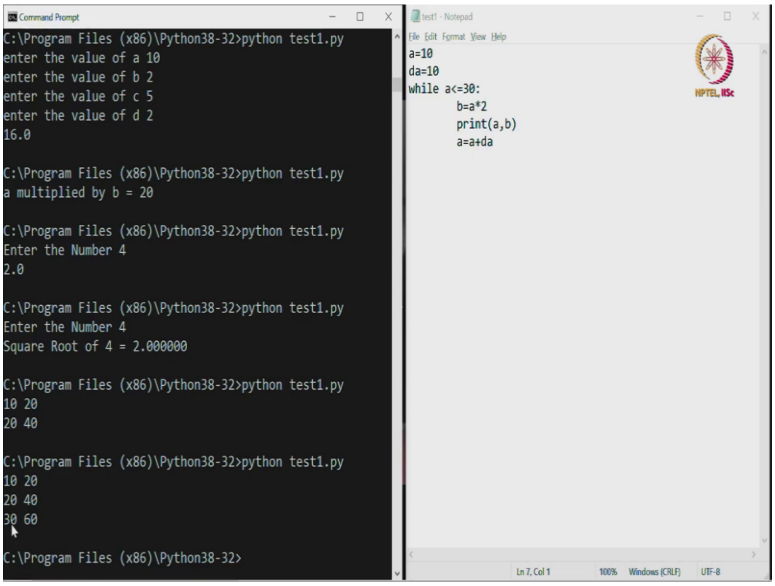
== equal to

!= not equal to

> greater than

< less than

Time dependent Quantum Chemistry



```
C:\Program Files (x86)\Python38-32>python test1.py
enter the value of a 10
enter the value of b 2
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>python test1.py
a multiplied by b = 20

C:\Program Files (x86)\Python38-32>python test1.py
Enter the Number 4
2.0

C:\Program Files (x86)\Python38-32>python test1.py
Enter the Number 4
Square Root of 4 = 2.000000

C:\Program Files (x86)\Python38-32>python test1.py
10 20
20 40

C:\Program Files (x86)\Python38-32>python test1.py
10 20
20 40
30 60

C:\Program Files (x86)\Python38-32>
```

```
a=10
da=10
while a<=30:
        b=a*2
        print(a,b)
        a=a+da
```

So, instead of repeating several times, there is a way of making it the making the execution, automate the execution using Python, and we will use two different loops for that. One is while loop and other one is for loop. While loop, we will go over the example of while loop first, we will take a look at it how the loop construction is given. A while loop is used to repeat a statement as long as a condition is true.

So, this loop is used as long as a condition is true. So, it will repeat whatever written in the loop and in Python, the loop instruction given the statement given in the loop is indented. You see there is a space we have given either you can use a space or tab and the loop heading with a condition is given followed by and in the end of the heading we have a colon sign which is defining the loop.

So, what happens when you start a value with 10 and increment 10 let us say, then when we start the while loop, it will check whether a value is less than equals 30. There are a number of Boolean true, false kind of operators we have in in Python for an example, this one indicates it is less than or equal to, less than or equal to this indicates equal to, then this indicates not equal to, this indicates greater than and this indicates less than.

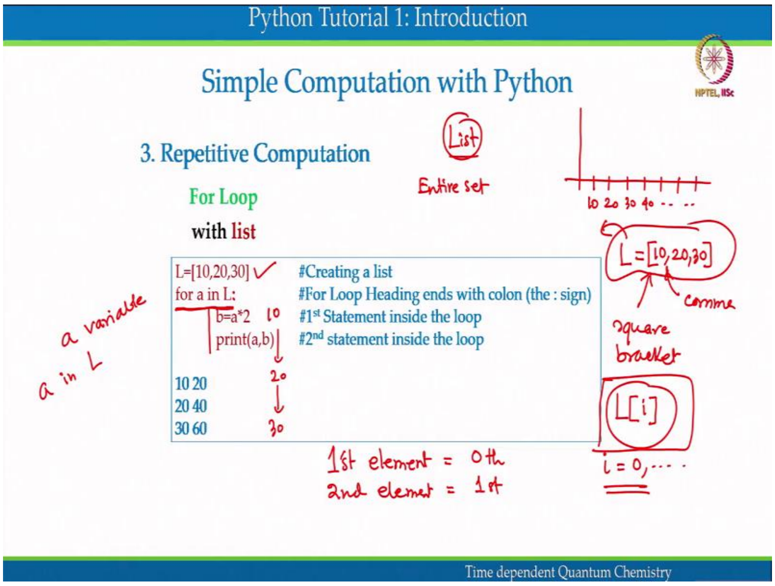
So, there are many Boolean expressions, one can look up in the Python documentation accordingly and what we are using is that is less than or equal expression. So, when entering into the loop, it will check what is the value of a if the value of a is less than equals 30, it will perform this task and, in the task, you are suggesting b equals a multiplied by 2, then you are it will be printing a, b and then it will update the value.

So, this is the earlier value and now this is the updated value and this updated value will be recheck whether the condition is fulfilled or not. If the condition is true, then it is going to rerun the loop and if the condition is false, then it will stop running it. So, let us look at it, in the laptop. We will start with a equals 10, then da that is the increment is also 10. Then, while a, I am defining the condition equals 30.

Then I am going to use a tab to define different statements given in the loop, a*2 then print a comma b and then update the value as a plus da. So, if we run the programme, we get 10, 20, 20, 40, 30, 60. So, you check that the loop has stopped, because 30 once the a value is 30, after incrementing, once a value is 30, it is checking the value and the moment it is becoming 30 it has run, but the moment it has become 40 the next run next iteration 40, it is becoming false, because a must be less than 30 or equal to 30 to run this iteration.

So, once a add ups 40 value, at the 40 value gone to the slide right now, at the 40 value, it is supposed to give me 80, but this iteration will not run because this condition becomes false and while loop will work as long as the condition is true.

(Refer Slide Time: 26:18)



Python Tutorial 1: Introduction

Simple Computation with Python

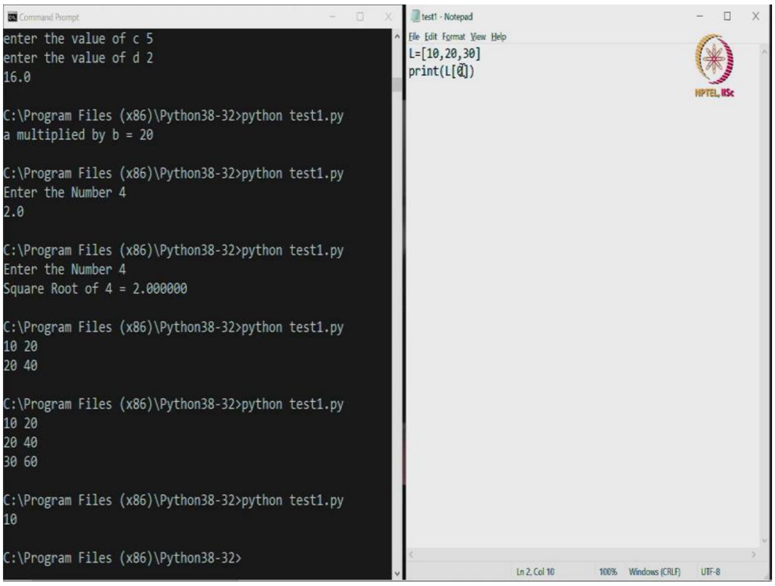
3. Repetitive Computation

For Loop

with list

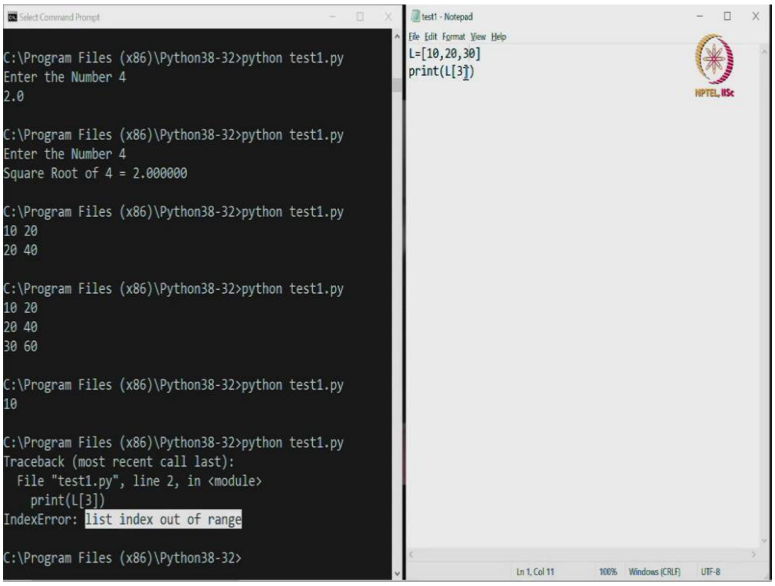
```
L=[10,20,30]        #Creating a list
for a in L:         #For Loop Heading ends with colon (the : sign)
     b=a*2          #1st Statement inside the loop
     print(a,b)     #2nd statement inside the loop

10 20
20 40
30 60
```

Time dependent Quantum Chemistry



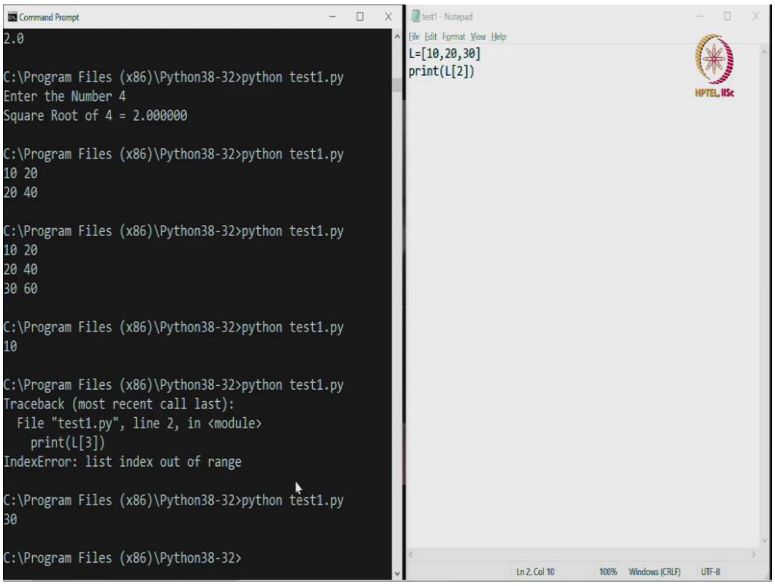
```
enter the value of c 5
enter the value of d 2
16.0

C:\Program Files (x86)\Python38-32>python test1.py
a multiplied by b = 20

C:\Program Files (x86)\Python38-32>python test1.py
Enter the Number 4
2.0

C:\Program Files (x86)\Python38-32>python test1.py
Enter the Number 4
Square Root of 4 = 2.000000

C:\Program Files (x86)\Python38-32>python test1.py
10 20
20 40

C:\Program Files (x86)\Python38-32>python test1.py
10 20
20 40
30 60

C:\Program Files (x86)\Python38-32>python test1.py
10

C:\Program Files (x86)\Python38-32>
```

```
L=[10,20,30]
print(L[0])
```



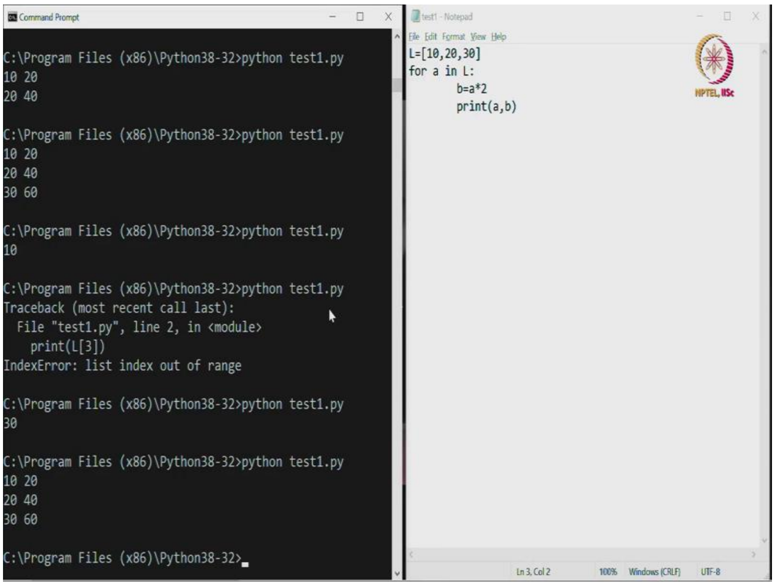
```
C:\Program Files (x86)\Python38-32>python test1.py
Enter the Number 4
2.0

C:\Program Files (x86)\Python38-32>python test1.py
Enter the Number 4
Square Root of 4 = 2.000000

C:\Program Files (x86)\Python38-32>python test1.py
10 20
20 40

C:\Program Files (x86)\Python38-32>python test1.py
10 20
20 40
30 60

C:\Program Files (x86)\Python38-32>python test1.py
10

C:\Program Files (x86)\Python38-32>python test1.py
Traceback (most recent call last):
  File "test1.py", line 2, in <module>
    print(L[3])
IndexError: list index out of range

C:\Program Files (x86)\Python38-32>
```

```
L=[10,20,30]
print(L[3])
```



```
2.0

C:\Program Files (x86)\Python38-32>python test1.py
Enter the Number 4
Square Root of 4 = 2.000000

C:\Program Files (x86)\Python38-32>python test1.py
10 20
20 40

C:\Program Files (x86)\Python38-32>python test1.py
10 20
20 40
30 60

C:\Program Files (x86)\Python38-32>python test1.py
10

C:\Program Files (x86)\Python38-32>python test1.py
Traceback (most recent call last):
  File "test1.py", line 2, in <module>
    print(L[3])
IndexError: list index out of range

C:\Program Files (x86)\Python38-32>python test1.py
30

C:\Program Files (x86)\Python38-32>
```

```
L=[10,20,30]
print(L[2])
```



```
C:\Program Files (x86)\Python38-32>python test1.py
10 20
20 40

C:\Program Files (x86)\Python38-32>python test1.py
10 20
20 40
30 60

C:\Program Files (x86)\Python38-32>python test1.py
10

C:\Program Files (x86)\Python38-32>python test1.py
Traceback (most recent call last):
  File "test1.py", line 2, in <module>
    print(L[3])
IndexError: list index out of range

C:\Program Files (x86)\Python38-32>python test1.py
30

C:\Program Files (x86)\Python38-32>python test1.py
10 20
20 40
30 60

C:\Program Files (x86)\Python38-32>
```

```
L=[10,20,30]
for a in L:
        b=a*2
        print(a,b)
```

We will now move to for loop and for, for loop we will introduce another functionality, which is built in functionality of Python that is listing a number, a list of numbers. So, one can prepare a list. For graph plotting, if we look at the x axis, we can have 10, 20, 30, 40 like this way, we can have the values for x, and the entire set can be considered to be this entire set can be considered to be a list in, in built in Python feature and this list name can be given anything, L, or it can be given M anything, but it will be specified within this square bracket and each number will be separated by comma.

So, I am specifying the list as 10, 20, 30 and that is the way we have written. Once we write down this way with a square bracket and each one is separated by comma each number in the list is separated by comma, that is the way we can prepare the list and once you prepare the list, one can in fact call a particular element of the list by this construct within this square bracket L i.

So, let us look at how does it work in my laptop. Let us say I have defined a list to be within square bracket 10, then 20, then 30. And I would like to print the first element of the list. How can I print? I can print it like this way L again square bracket first element is given by zeroth element because index starting with 0. So, I have defined the list already, and I am just calling to print L zeroth element and is giving me 10.

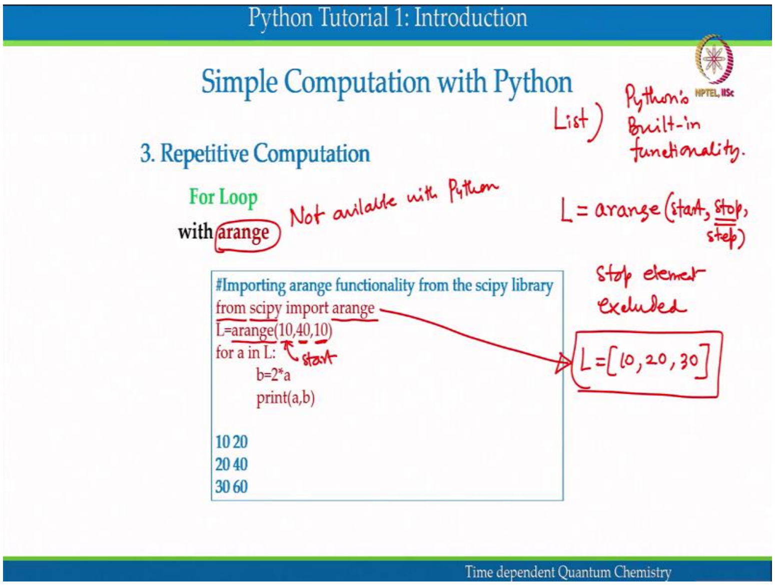
The first element. So, I will write down here if I define a list with this the first element, first element in the list is going to be zeroth element. Then second element of the list is the first one. So, i is actually varying from the index is varying from 0, and So, on. So, we will move to this laptop right now. If I change to 3, what I get, I get an error because it is out of range it is showing that the list index is out of range.

Why? because I do not have the fourth element L 3 defines the fourth element in the list and I have only three elements. So, the last element is going to be second i equals 2, and then it will print 30. So, this is the way one can define a list. So, to define a list, I have to use this construct this name can be anything and if I want to call a particular element from the list, this is the construct we should use where index is starting from 0.

So, what we have done in the for loop, we have defined the list and the for loop construction again it ends with the heading ends with a colon that we remember and then indented part that is the there is showing the iteration part, but the construction is for a in L, a is a variable. So, a in L, it means that the element in L and it will start from the first element. So, b equals a multiplied by 2 in the first iteration a will be taken to be 10. In the second iteration a will be taken to be 20 and third iteration a will be taking to be 30.

After that there is no element that is why for loop will end. So, what is going on in the for loop? It is actually picking up one by one element from the list. So, I have this list prepared and then I am going to define the variable a in L with a colon then within tab, b equals a multiplied by 2 and then print a comma b. This for loop will run as long as I have the elements in the list. So, if we run it we get 10, 20, 20, 40, 30, 60. That is the list we wanted to prepare and it is showing up.

(Refer Slide Time: 32:20)



Python Tutorial 1: Introduction

Simple Computation with Python

List) Python's Built-in functionality.

3. Repetitive Computation

For Loop

with arange

Not available with Python

L = arange (start, stop, step)

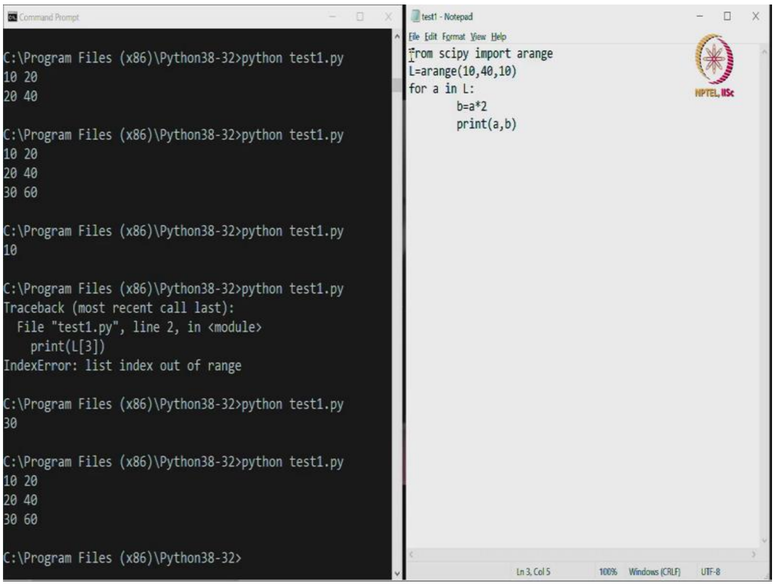
```
#Importing arange functionality from the scipy library
from scipy import arange
L=arange(10,40,10)
for a in L:
        b=2*a
        print(a,b)

10 20
20 40
30 60
```

start

Stop element excluded

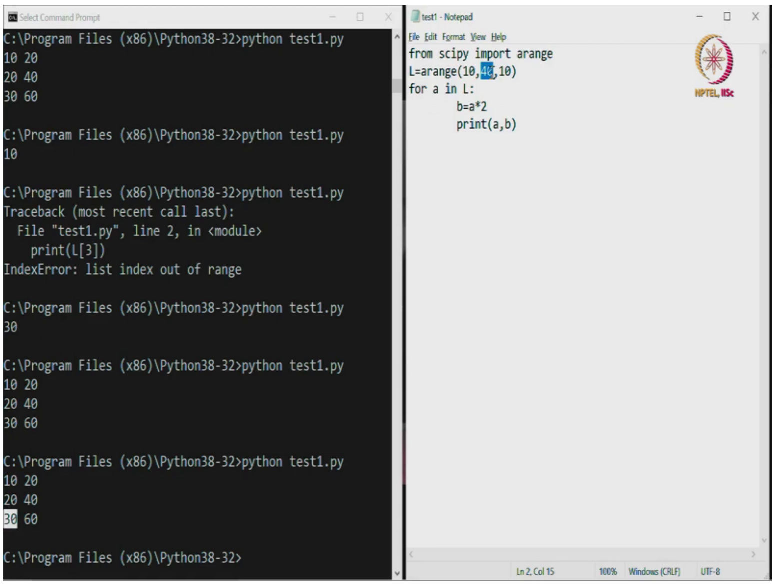
L=[10,20,30]

Time dependent Quantum Chemistry



```
C:\Program Files (x86)\Python38-32>python test1.py
10 20
20 40

C:\Program Files (x86)\Python38-32>python test1.py
10 20
20 40
30 60

C:\Program Files (x86)\Python38-32>python test1.py
10

C:\Program Files (x86)\Python38-32>python test1.py
Traceback (most recent call last):
  File "test1.py", line 2, in <module>
    print(L[3])
IndexError: list index out of range

C:\Program Files (x86)\Python38-32>python test1.py
30

C:\Program Files (x86)\Python38-32>python test1.py
10 20
20 40
30 60

C:\Program Files (x86)\Python38-32>
```

```
from scipy import arange
L=arange(10,40,10)
for a in L:
        b=a*2
        print(a,b)
```



```
C:\Program Files (x86)\Python38-32>python test1.py
10 20
20 40
30 60

C:\Program Files (x86)\Python38-32>python test1.py
10

C:\Program Files (x86)\Python38-32>python test1.py
Traceback (most recent call last):
  File "test1.py", line 2, in <module>
    print(L[3])
IndexError: list index out of range

C:\Program Files (x86)\Python38-32>python test1.py
30

C:\Program Files (x86)\Python38-32>python test1.py
10 20
20 40
30 60

C:\Program Files (x86)\Python38-32>python test1.py
10 20
20 40
30 60

C:\Program Files (x86)\Python38-32>
```

```
from scipy import arange
L=arange(10,40,10)
for a in L:
        b=a*2
        print(a,b)
```

We will check another way of preparing the list. This list is Python's built-in functionality. I can use that, on the other hand, there is another way to prepares a list. In fact, more specifically is going to be an array but will not discuss the meaning of array immediately we will consider as a list with the help of arange this is not arrange this is called arange and this arange is not available with Python.

So, we have to import it, and importing means we will be importing from scipy. So, from scipy we will first import this arange functionality, this arange functionality works like this way again I have to give the name for this list. List equals arange, range, within bracket start comma stop comma increment and or step. This is what we have written. But we have to remember when is preparing this list it includes all the elements except for stop. So, stop element excluded in the list.

So, let us look at this; L equals arange 10, 40, 10 what does it mean? This is the starting point, so, I will start, so, L is going to be then 10, then it will, it should go, stop at 40 and with an increment 10. So, if I have increment 10 so, next element is going to be 20, next element will be 30, next element is going to be 40 but it will exclude 40 and it will not include it in the list.

So, this is the list I am going to prepare with the help of arange functionality. The reason why I am using this arange functionality, this functional tool be used repeatedly in this course to define one dimensional array and what is the meaning off array we will not think about it right now we will consider it as a list of numbers we are preparing. And when you are using arange functional you should again remember that the stop is excluded from the list.

Remaining part is the same so we will do that, we will first import from scipy, the construct is from scipy import arange functionality, then I use this functionality to define the list arrange. Starting point 10, ending point 40, increment 10, 40 will be excluded from the list. Then remaining part for loop remains to be the same.

If I execute the programme, I get 10, 20, 20, 40, 30, 60. Look at this 10, 20 and 30 here 10, 20 and 30 has been included in the list, but 40 has been excluded from the list. So, stop point will be excluded from the list, from the list if we use this arange functionality. We will stop here and we will continue this tutorial in the next session.