## Report

## ON

## Summer Training June, 2016

## BY

**(MANISHA KUMARI)**

**(2K15CSUN01080)**



**MANAV RACHNA UNIVERSITY**

**Established, vide Haryana Act no.26 of 2014**

**(Formerly Manav Rachna College of Engineering,**

**NAAC Accredited ‘A’ Grade Institute)**

## DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

## MANAV RACHNA UNIVERSITY

## FARIDABAD, HARYANA (INDIA)

## JUNE, 2016

**INDEX:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENT** |  |
| **MODULE 1** | | |
| **1.** | **Software Testing** |  |
|  | **Conclusion** |  |
|  |  |  |
| **MODULE 2** | | |
| **2.** | **Python** |  |
|  | **Conclusion** |  |

**Acknowledgement**

I **MANISHA KUMARI** have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to **Manav Rachna University(MRU)** ,Faridabad for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express my gratitude towards **HOD Mrs. Hanu Bharwdwaj** & **Teachers of CST Department MRU** for their kind co-operation and encouragement which help me in completion of this project.

I would like to express my special gratitude and thanks to **JLJ India & Skytesters** persons for giving me such attention and time.

.

**Date Of Submission-24/06/16** **Faculty Signature**

**Software Testing**

Software testing is a process of executing a program or application with the intent of finding the software bugs.

* It can also be stated as the process of [validating](http://istqbexamcertification.com/what-is-validation-in-software-testing-or-what-is-software-validation/) and [verifying](http://istqbexamcertification.com/what-is-verification-in-software-testing-or-what-is-software-verification/) that a software program or application or product:
* Meets the business and technical requirements that guided it’s design and development
* Works as expected
* Can be implemented with the same characteristic.

Let’s break the definition of [Software testing](http://istqbexamcertification.com/what-is-a-software-testing/) into the following parts:

1)  Process:  Testing is a process rather than a single activity.

2)  All Life Cycle Activities: Testing is a process that’s take place throughout the[Software Development Life Cycle (SDLC)](http://istqbexamcertification.com/what-are-the-software-development-life-cycle-sdlc-phases/).

* The process of designing tests early in the life cycle can help to prevent defects from being introduced in the code. Sometimes it’s referred as“verifying the test basis via the [test design](http://istqbexamcertification.com/what-is-test-design-or-how-to-specify-test-cases/)”.
* The test basis includes documents such as the requirements and design specifications.

3)  [Static Testing](http://istqbexamcertification.com/what-is-static-testing/):  It can test and find defects without executing code. Static Testing is done during verification process. This testing includes reviewing of the documents (including source code) and static analysis. This is useful and cost effective way of testing.  For example: reviewing,[walkthrough](http://istqbexamcertification.com/what-is-walkthrough-in-software-testing/), [inspection](http://istqbexamcertification.com/what-is-inspection-in-software-testing/), etc.

4)  [Dynamic Testing](http://istqbexamcertification.com/what-is-dynamic-testing-technique/):  In dynamic testing the software code is executed to demonstrate the result of running tests. It’s done during validation process. For example: [unit testing](http://istqbexamcertification.com/what-is-unit-testing/),[integration testing](http://istqbexamcertification.com/what-is-integration-testing/), [system testing](http://istqbexamcertification.com/what-is-system-testing/), etc.

 5)  [Planning](http://istqbexamcertification.com/what-is-the-purpose-and-importance-of-test-plans/):  We need to plan as what we want to do. We control the test activities, we report on testing progress and the status of the software under test.

6)  Preparation:  We need to choose what testing we will do, by selecting test conditions and [designing test cases](http://istqbexamcertification.com/what-is-test-design-or-how-to-specify-test-cases).

7)  Evaluation:  During evaluation we must check the results and evaluate the software under test and the completion criteria, which helps us to decide whether we have finished testing and whether the software product has passed the tests.

**Performance Testing**

Software Performance testing is type of testing perform to determine the performance of system to major the measure, validate or verify quality attributes of the system like responsiveness, Speed, Scalability, Stability under variety of load conditions. The system is tested under a mixture of load conditions and check the time required responding by the system under varying workloads. Software performance testing involves the testing of application under test to ensure that application is working as expected under variety of load conditions. The goal of performance testing is not only find the bugs in the system but also eliminate the performance bottlenecks from the system.

**Requirement of Performance Testing**

Before going live in the market, the software system should be tested against the Speed, Stability and scalability under variety of load conditions. If system goes live without doing performance testing may cause the issues like running system slow while simultaneously accessing system by several users, poor usability which likely to gain the bad reputation and it affects the expected sales goal directly. Performance testing encompasses a range of different tests which enable analysis of various aspects of the system. The Performance testing is tells about what needs to fix before going live (mainly the issues faced under the variety of load conditions).

**Various Techniques of performance Testing**

**Load testing**

**Stress testing**

* **Soak testing**
* **Spike testing**

**Process Of Performance Testing**

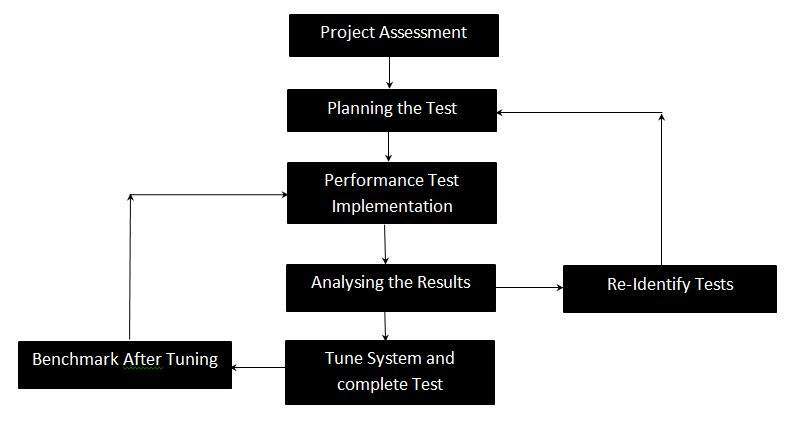


Fig 1- Process of Performance Testing

**Attributes to Performance Testing:**

* Speed
* Scalability
* Stability
* reliability

**Types Of Performance Testing:**



Fig 2: Types of Performance Testing

## 1) Load Testing:

Load Testing is type of performance testing to check system with constantly increasing the load on the system until the time load is reaches to its threshold value. Here Increasing load means increasing number of concurrent users, transactions & check the behavior of application under test. It is normally carried out underneath controlled environment in order to distinguish between two different systems. It is also called as “Endurance testing” and “Volume testing”. The main purpose of load testing is to monitor the response time and staying power of application when system is performing well under heavy load. Load testing comes under the[Non Functional Testing](http://www.softwaretestingclass.com/functional-testing-vs-non-functional-testing/) & it is designed to test the non-functional requirements of a software application.

Load testing is perform to make sure that what amount of load can be withstand the application under test. The successfully executed load testing is only if the specified test cases are executed without any error in allocated time.

Simple examples of load testing:

* Testing printer by sending large job.
* Editing a very large document for testing of word processor
* Continuously reading and writing data into hard disk.
* Running multiple applications simultaneously on server.
* Testing of mail server by accessing thousands of mailboxes
* In case of zero-volume testing & system fed with zero load.

## 2) Stress Testing:

Stress Testing is performance testing type to check the stability of software when hardware resources are not sufficient like CPU, memory, disk space etc.

Stress testing is Negative testing where we load the software with large number of concurrent users/processes which cannot be handled by the systems hardware resources. This testing is also known as Fatigue testing, this testing should capture the stability of the application by testing it beyond its bandwidth capacity.

The main idea behind stress testing is to determine the failure of system and to keep an eye on how the system gracefully get recover back, this quality is known as recoverability. Stress testing comes under the [Non Functional Testing](http://www.softwaretestingclass.com/functional-testing-vs-non-functional-testing/) & it is designed to test the non-functional requirements of a software application. This testing is to be carried out under controlled environment before launch, so that we can accurately capture the system behavior under most erratic scenarios.

## 3) Spike Testing:

Spike testing is subset of Stress Testing. A spike test is carried out to validate the performance characteristics when the system under test subjected to workload models and load volumes that repeatedly increase beyond anticipated production operations for short periods of time.

## 4) Endurance Testing:

Endurance testing is a non functional type of testing. Endurance testing involves testing a system with a expected amount of load over a long period of time to find the behavior of system. Let’s take a example where system is designed to work for 3 hrs of time but same system endure for 6 hrs of time to check the staying power of system. Most commonly test cases are executed to check the behavior of system like memory leaks or system fails or random behavior. Sometimes endurance testing is also referred as Soak testing.

## 5) Scalability Testing:

Scalability Testing is type of non-functional tests and it is the testing of a software application for determine its capability to scale up in terms of any of its non-functional capability like the user load supported, the number of transactions, the data volume etc. The main aim if this testing is to understand at what peak the system prevent more scaling.

## 6) Volume Testing:

Volume testing is non-functional testing which refers to testing a software application with a large amount of data to be processed to check the efficiency of the application. The main goal of this testing is to monitor the performance of application under varying database volumes.

# Top Performance Testing Tools:

* [Apache JMeter](https://jmeter.apache.org/download_jmeter.cgi)
* [WebLOAD](http://www.radview.com/)
* [LoadRunner](http://www8.hp.com/in/en/software-solutions/software.html?compURI=1175451)
* [NeoLoad](http://go.neotys.com/NeoLoad-Trial-Download.html)
* [LoadUI](http://www.loadui.org/)
* [OpenSTA](http://opensta.org/download.html)
* [WAPT](http://www.loadtestingtool.com/download.shtml)
* [LoadImpact](http://loadimpact.com/)
* [Loadster](http://www.loadsterperformance.com/loadster/workbench)
* [Httperf](http://www.hpl.hp.com/research/linux/httperf/download.php)
* [Rational Performance Tester](http://www.ibm.com/developerworks/downloads/r/rpt/)
* [QEngine (ManageEngine)](http://www.manageengine.com/products/qengine/)
* [Testing Anywhere](http://www.automationanywhere.com/Testing/Downloads/Testing-Anywhere-Setup750.exe)
* [CloudTest](http://www.soasta.com/products/cloudtest-performance/)
* [Loadstorm](http://loadstorm.com/)

# Common Performance Problems:

In the software testing of an application Speed is one of the important attribute. User will not happy to work with slow system. The performance testing uncovers the performance bottlenecks & defects to maintain interest and attention of user. Here is the list of most commonly performance problems observed in software system:

* Poor response time
* Long Load time
* Bottlenecking
* Poor scalability
* Software configuration issues (for the Web server, load balancers, databases etc.)
* Disk usage
* Operating System limitations
* Poor network configuration
* Memory utilization
* CPU utilization
* Insufficient hardware resources

.

**APACHAE JMETER**

**Apache JMeter** is an [Apache](https://en.wikipedia.org/wiki/Apache_Software_Foundation) [project](https://en.wikipedia.org/wiki/Project) that can be used as a [load testing](https://en.wikipedia.org/wiki/Load_testing) tool for analyzing and measuring the performance of a variety of services, with a focus on [web applications](https://en.wikipedia.org/wiki/Web_application).

JMeter can be used as a unit-test tool for JDBC database connections,FTP,LDAP,Webservices , JMS, HTTP, generic TCP connections and OS native processes. One can also configure JMeter as a monitor, although this is typically considered *ad hoc* rather than advanced monitoring. It can be used for some functional testing as well.

JMeter supports variable parametrization, assertions (response validation), per-thread cookies, configuration variables and a variety of reports.

JMeter architecture is based on [plugins](https://en.wikipedia.org/wiki/Plugins). Most of its "out of the box" features are implemented with plugins. Off-site developers can easily extend JMeter with custom plugins.

**1.Test Plan:**

A test plan describes a series of steps JMeter will execute when run. A complete test plan will consist of one or more Thread Groups, logic controllers, sample generating controllers, listeners, timers, assertions, and configuration elements.

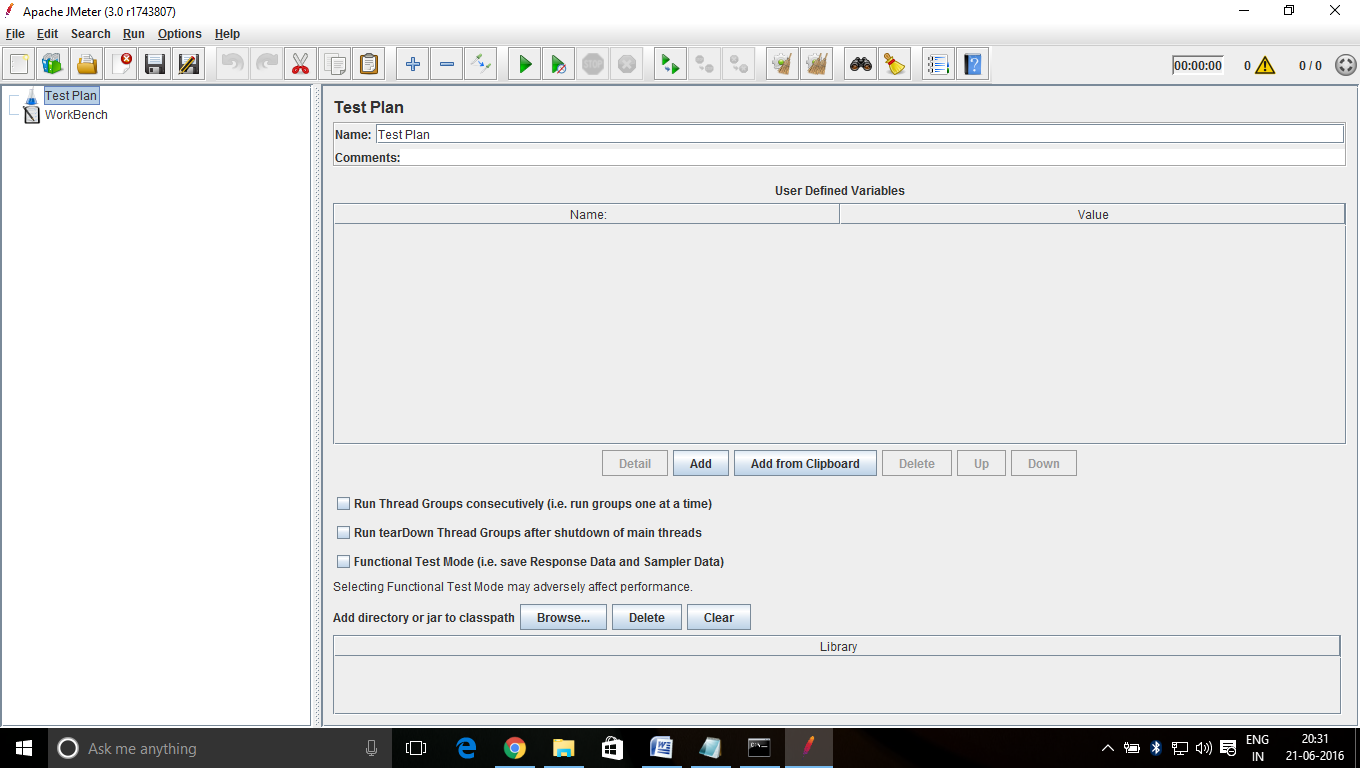


Fig 3: Apachae Jmeter Window having Testplan open.

**1.1 Thread Group:**

Thread group elements are the beginning points of any test plan. All controllers and samplers must be under a thread group. Other elements, e.g. Listeners, may be placed directly under the test plan, in which case they will apply to all the thread groups. As the name implies, the thread group element controls the number of threads JMeter will use to execute your test. The controls for a thread group allow you to:

* Set the number of threads
* Set the ramp-up period
* Set the number of times to execute the test

Each thread will execute the test plan in its entirety and completely independently of other test threads. Multiple threads are used to simulate concurrent connections to your server application.

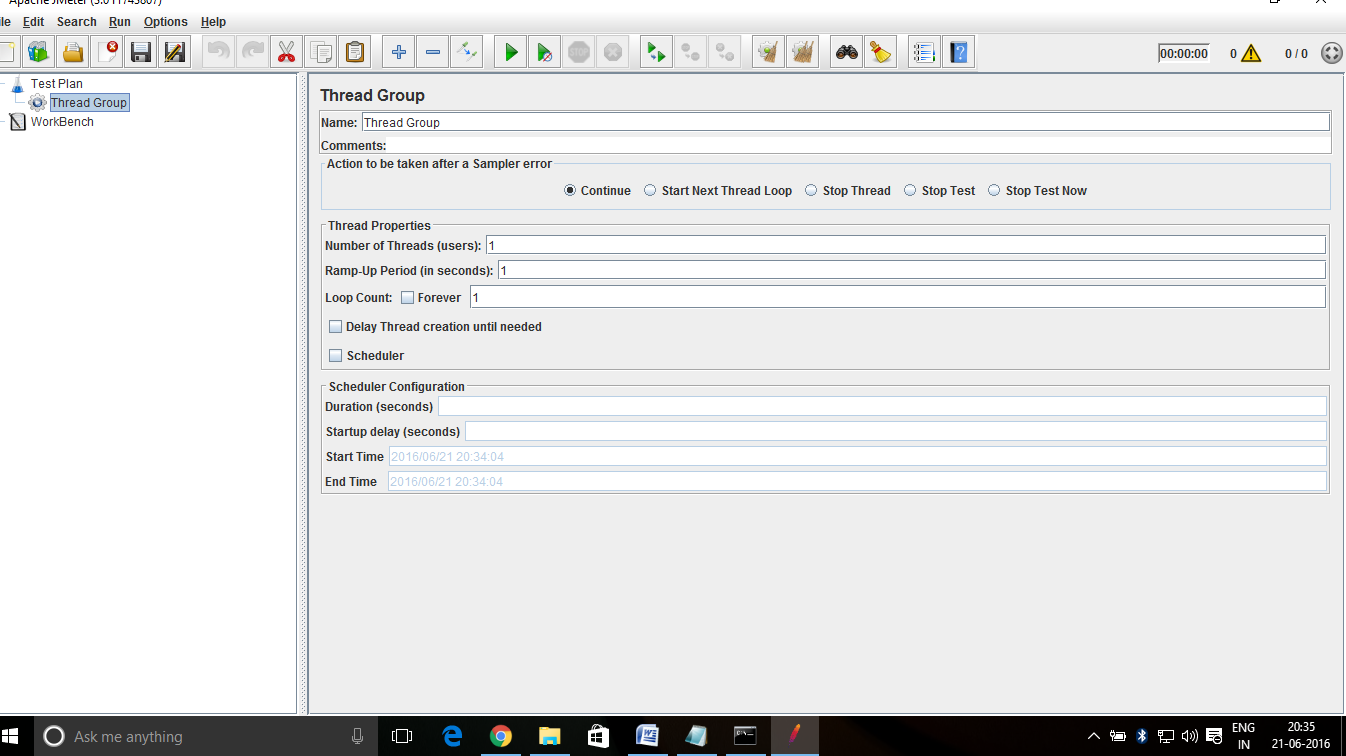


Fig 4:Showing Thread Group Window

**1.1.1Logic Controllers:**

Logical Controllers let you customize the logic that JMeter uses to decide when to send requests. For example, you can add an Interleave Logic Controller to alternate between two HTTP Request Samplers. For more information, see [Logical Controllers](#logic_controller).

1.1.2 **Configuration Elements**:

A configuration element works closely with a Sampler. Although it does not send requests (except for[HTTP(S) Test Script Recorder](#HTTP(S)_Test_Script_Recorder)), it can add to or modify requests.

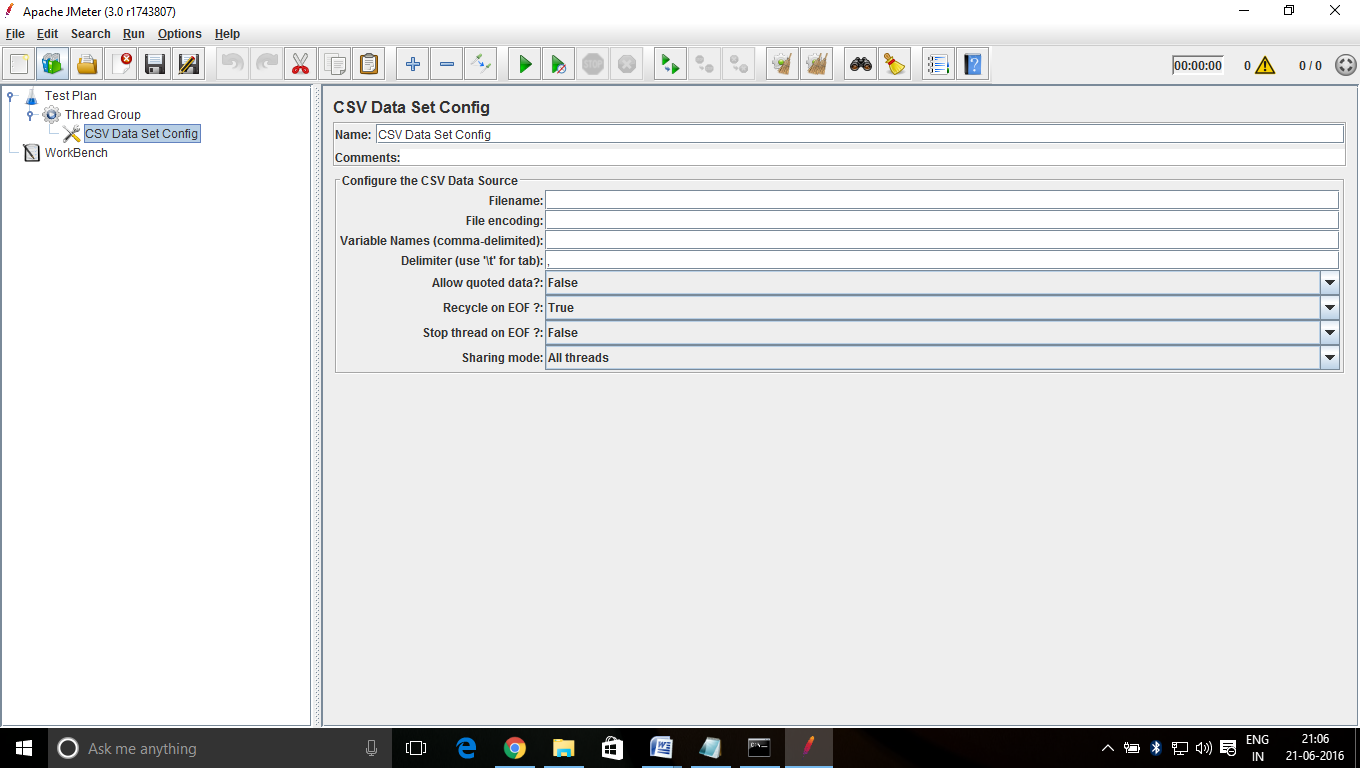


Fig 5: Showing CSV Data set Config(Configuration Elements)

**1.1.3 Timers:**

By default, a JMeter thread executes samplers in sequence without pausing. We recommend that you specify a delay by adding one of the available timers to your Thread Group. If you do not add a delay, JMeter could overwhelm your server by making too many requests in a very short amount of time. A timer will cause JMeter to delay a certain amount of time before each sampler which is in its [scope](#scoping_rules).

**1.1.4 Pre-Processors :**

A Pre-Processor executes some action prior to a Sampler Request being made. If a Pre-Processor is attached to a Sampler element, then it will execute just prior to that sampler element running. A Pre-Processor is most often used to modify the settings of a Sample Request just before it runs, or to update variables that aren't extracted from response text. See the [scoping rules](#scoping_rules) for more details on when Pre-Processors are executed.

**1.1.5 Samplers :**

Samplers tell JMeter to send requests to a server and wait for a response. They are processed in the order they appear in the tree. Controllers can be used to modify the number of repetitions of a sampler.

JMeter samplers include:

* FTP Request
* HTTP Request (can be used for SOAP or REST Webservice also)
* JDBC Request
* Java object request
* JMS request
* JUnit Test request
* LDAP Request
* Mail request
* OS Process request
* TCP request

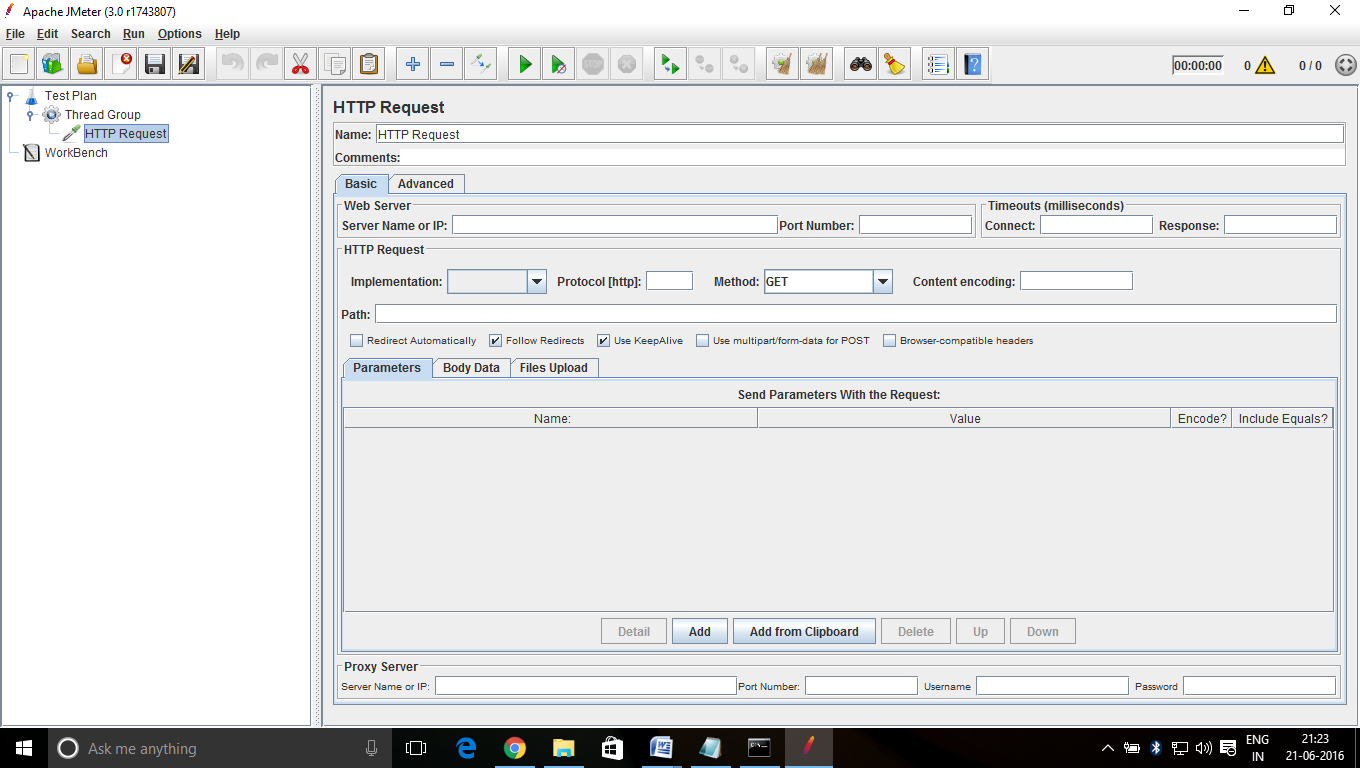


Fig 6: Showing HTTP request(Samplers)

**1.1.6 Post-Processor:**

A Post-Processor executes some action after a Sampler Request has been made. If a Post-Processor is attached to a Sampler element, then it will execute just after that sampler element runs. A Post-Processor is most often used to process the response data, often to extract values from it. See the [scoping rules](#scoping_rules) for more dertionsetails on when Post-Processors are executed.

**1.1.7 Assertions:**

Assertions allow you to assert facts about responses received from the server being tested. Using an assertion, you can essentially "test" that your application is returning the results you expect it to.

For instance, you can assert that the response to a query will contain some particular text. The text you specify can be a Perl-style regular expression, and you can indicate that the response is to contain the text, or that it should match the whole response

**1.1.8 Listener:**

Listeners provide access to the information JMeter gathers about the test cases while JMeter runs. The[Graph Results](#Graph_Results) listener plots the response times on a graph. The "View Results Tree" Listener shows details of sampler requests and responses, and can display basic HTML and XML representations of the response. Other listeners provide summary or aggregation information

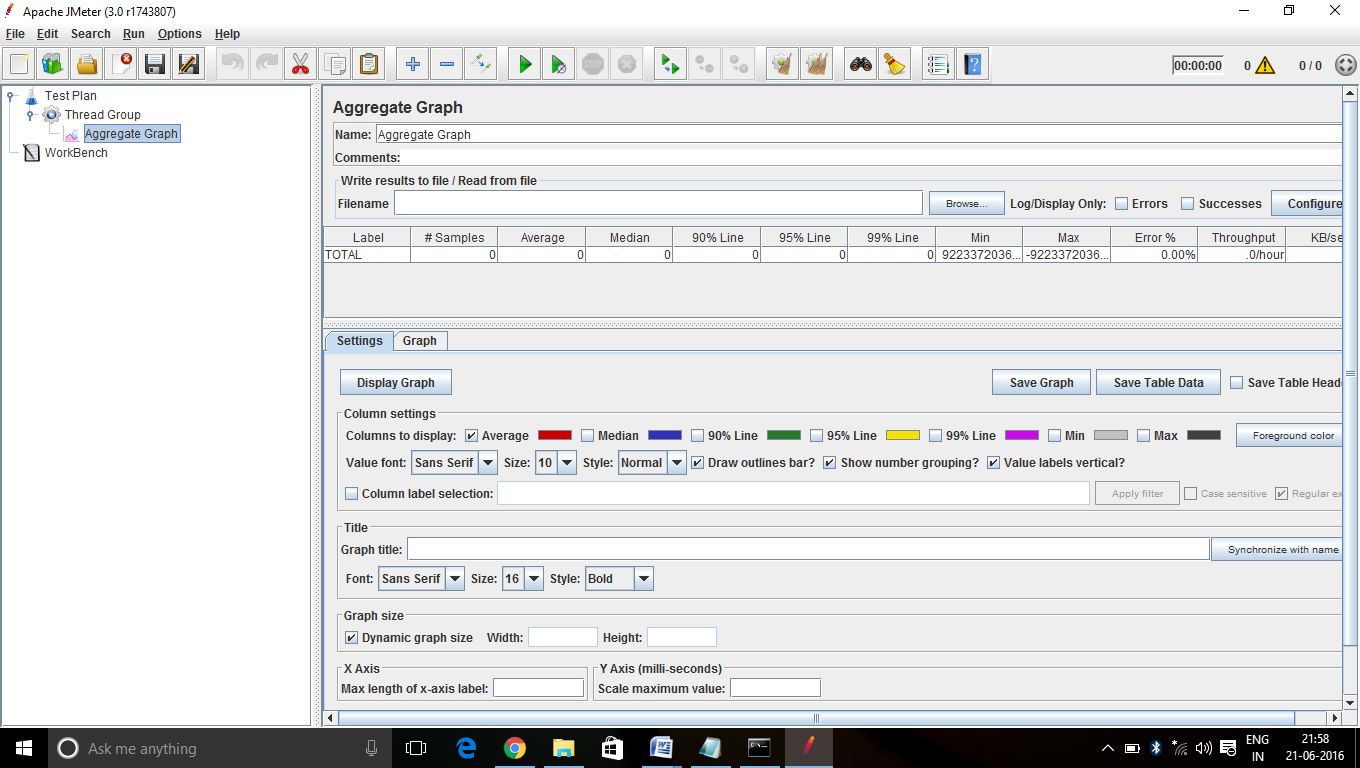


Fig 7 : Showing Aggregrate Graph (Listener)

**2.Workbench:**

The WorkBench simply provides a place to temporarily store test elements while not in use, for copy/paste purposes, or any other purpose you desire. When you save your test plan, WorkBench items are not saved with it by default unless you check "Save Workbench" option. Your WorkBench can be saved independently, if you like (right-click on WorkBench and choose Save).

Certain test elements are only available on the WorkBench:

* [HTTP(S) Test Script Recorder](#HTTP(S)_Test_Script_Recorder)
* [HTTP Mirror Server](#HTTP_Mirror_Server)
* [Property Display](#Property_Display)

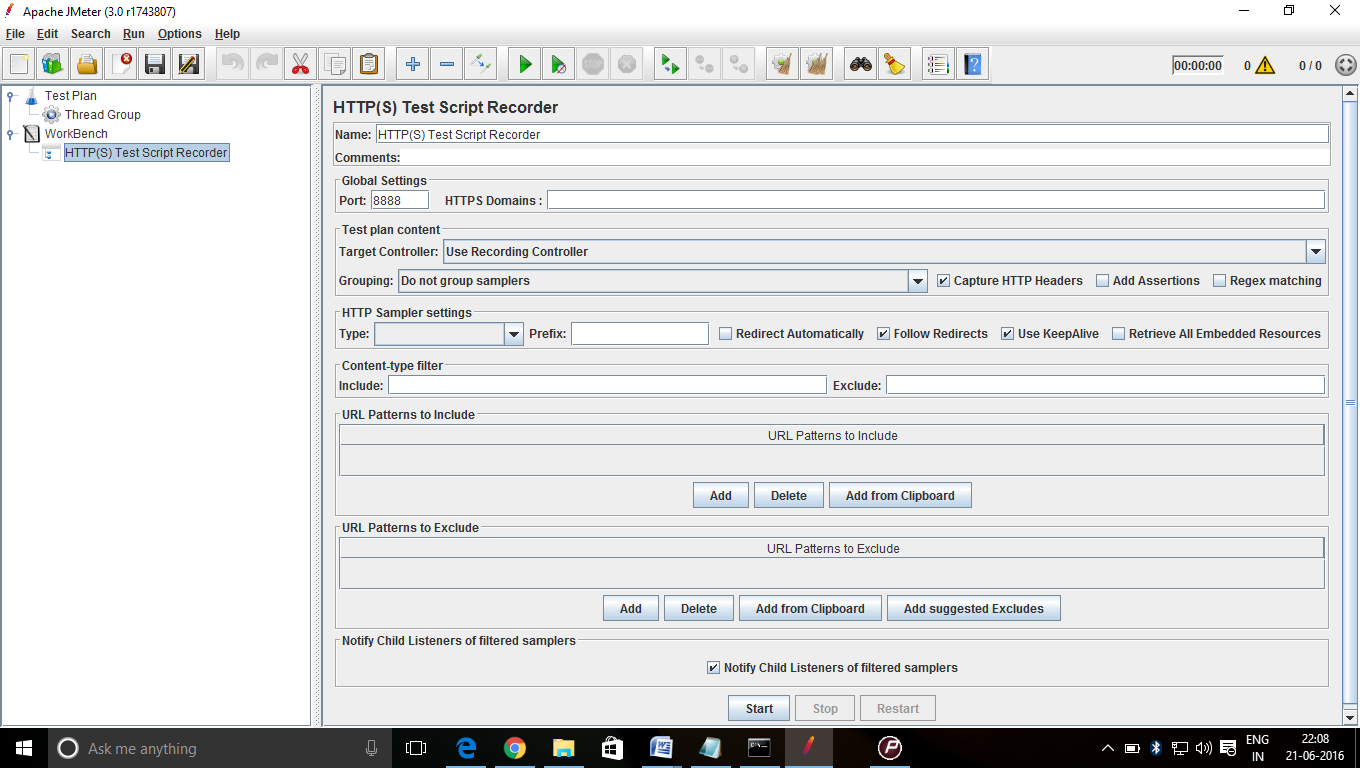


Fig 8: Showing HTTP test script recorder(Workbench)

**Performance Monitor*:***

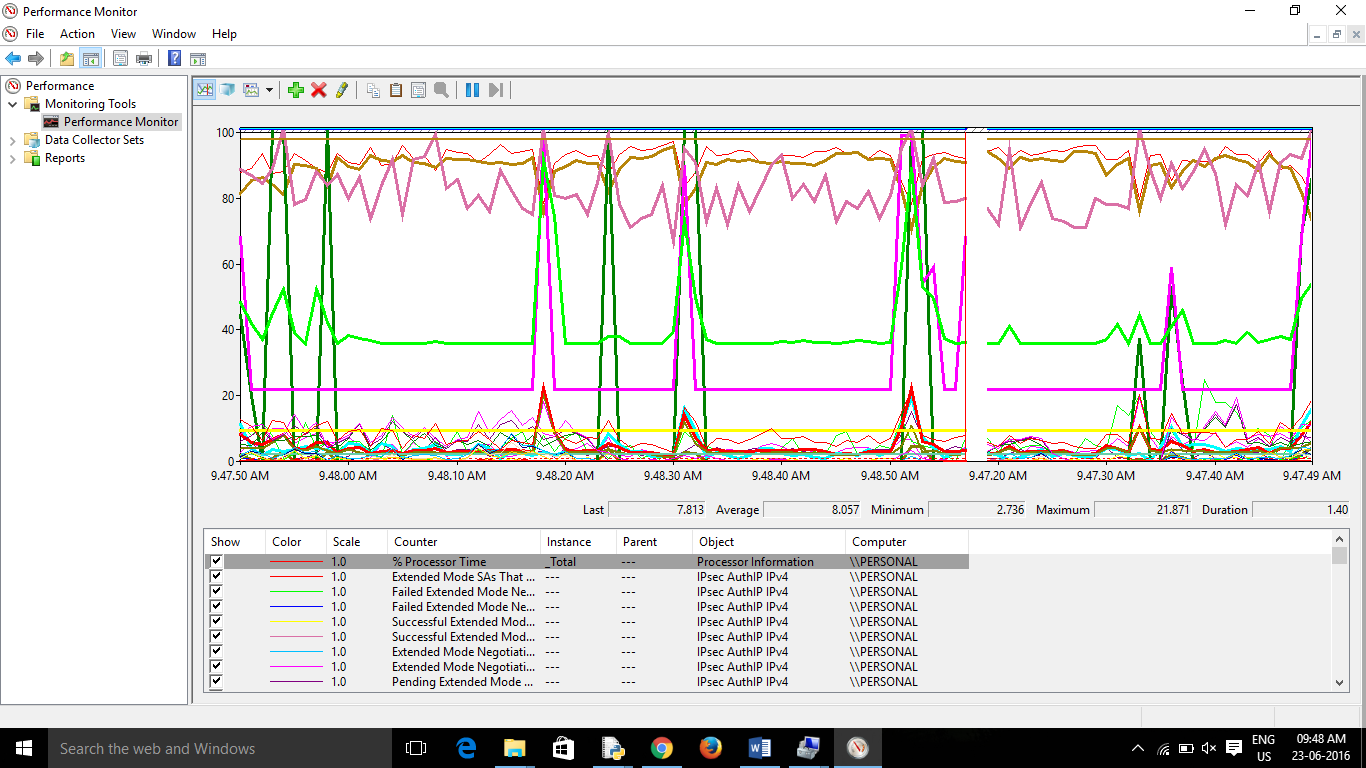
This performance monitor is available in every latest version of Microsoft Windows. The performance monitor is used to measure how the Components of the Windows are working.

**It consists of 3 parts-**

1) Monitoring Tool,

2) Data collector Sets

3) Reports



**How to ADD more module in Performance Monitor:**

* + - Click on the **+** Icon in or press Ctrl + N.
    - Now scroll in the Available counters.
    - Select all the Process you want to add to the report.
    - Now click on Add >> to add them to performance monitor.
    - Click ok & you are done with it.

**Reporting a Bug at app.skytesters.com**

1. Sign up at app.skytesters.com to create a new account

.

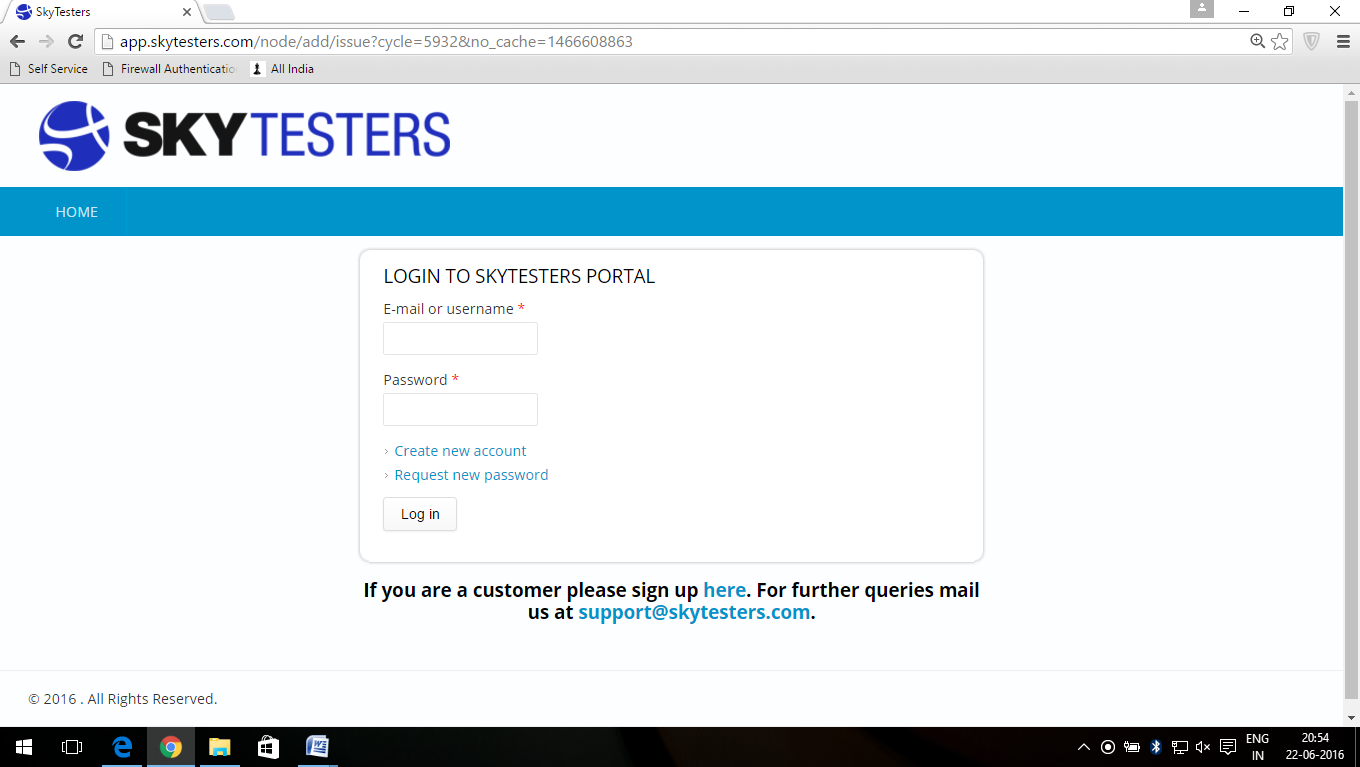


Fig 9: Shows the login page of skytesters

2. Register your Mobile device providing the details.

3. After logging in, select a test cycle.

4. Download the related app from the link provided in the test cycle.

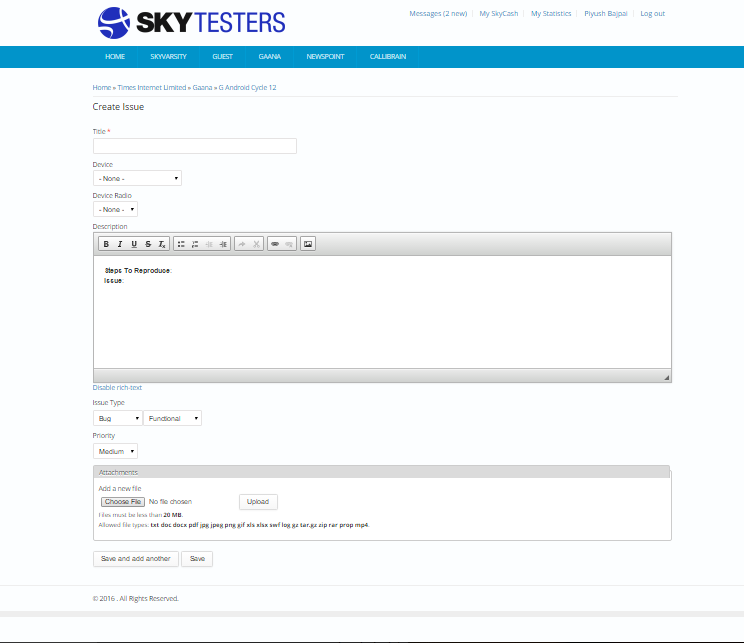
5. Install the app on your device s and start finding bugs in it.

6. Whenever a bug is encountered, open a MX Log Collector App to take a record of it.

7. Save the log in a text file.

8. Take a screenshot of the bug you encountered.

9. Click on the link “Report an Issue” on a test cycle page ,a new page will appear

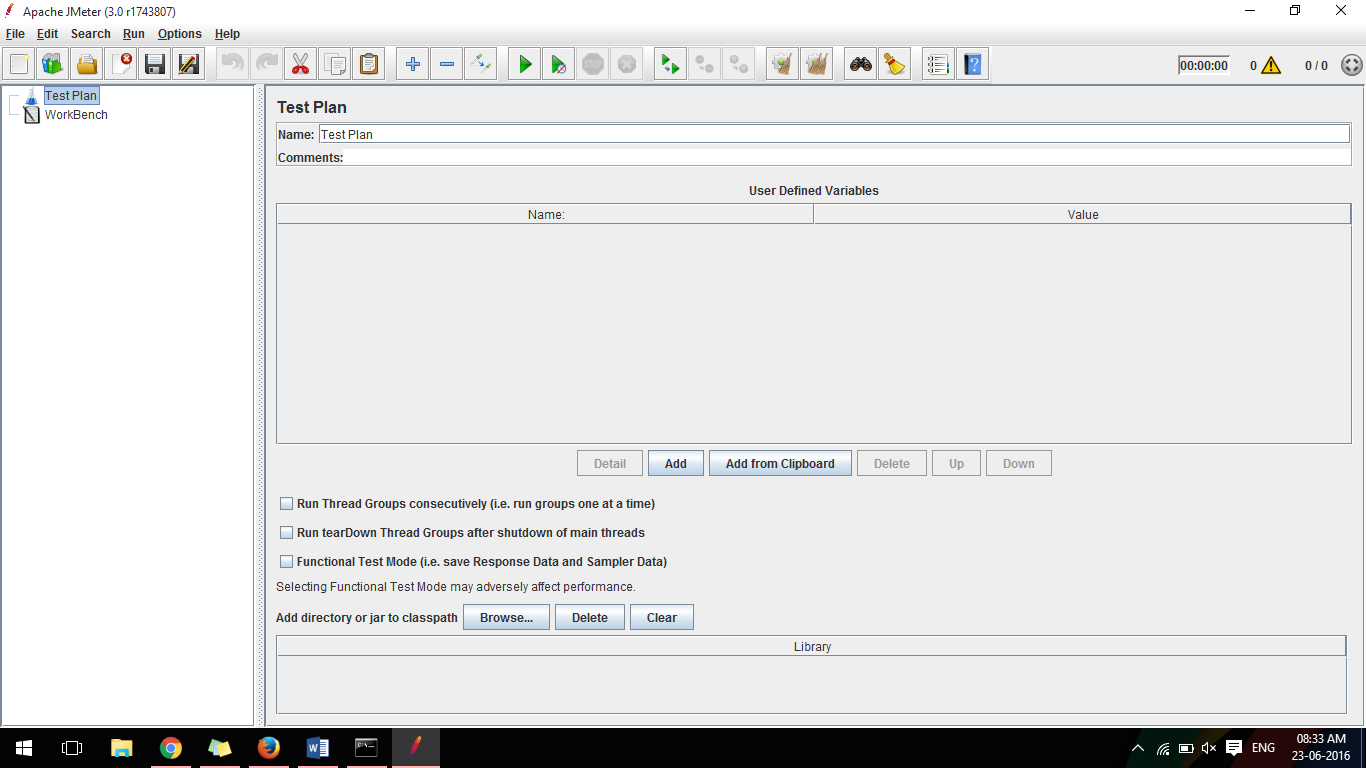
.

10. Following is the display of the page where you can successfully report your bug;

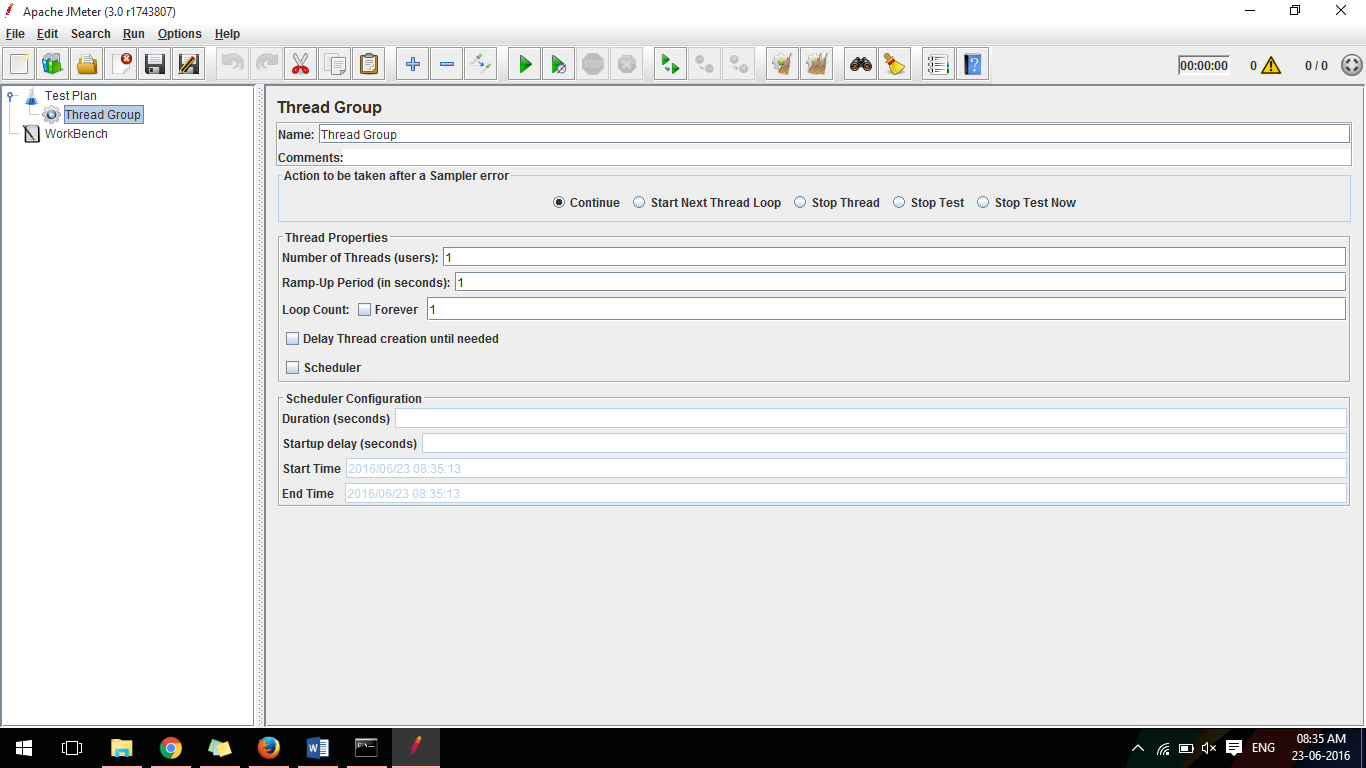
**NPTEL**

**NPTEL Project: -**

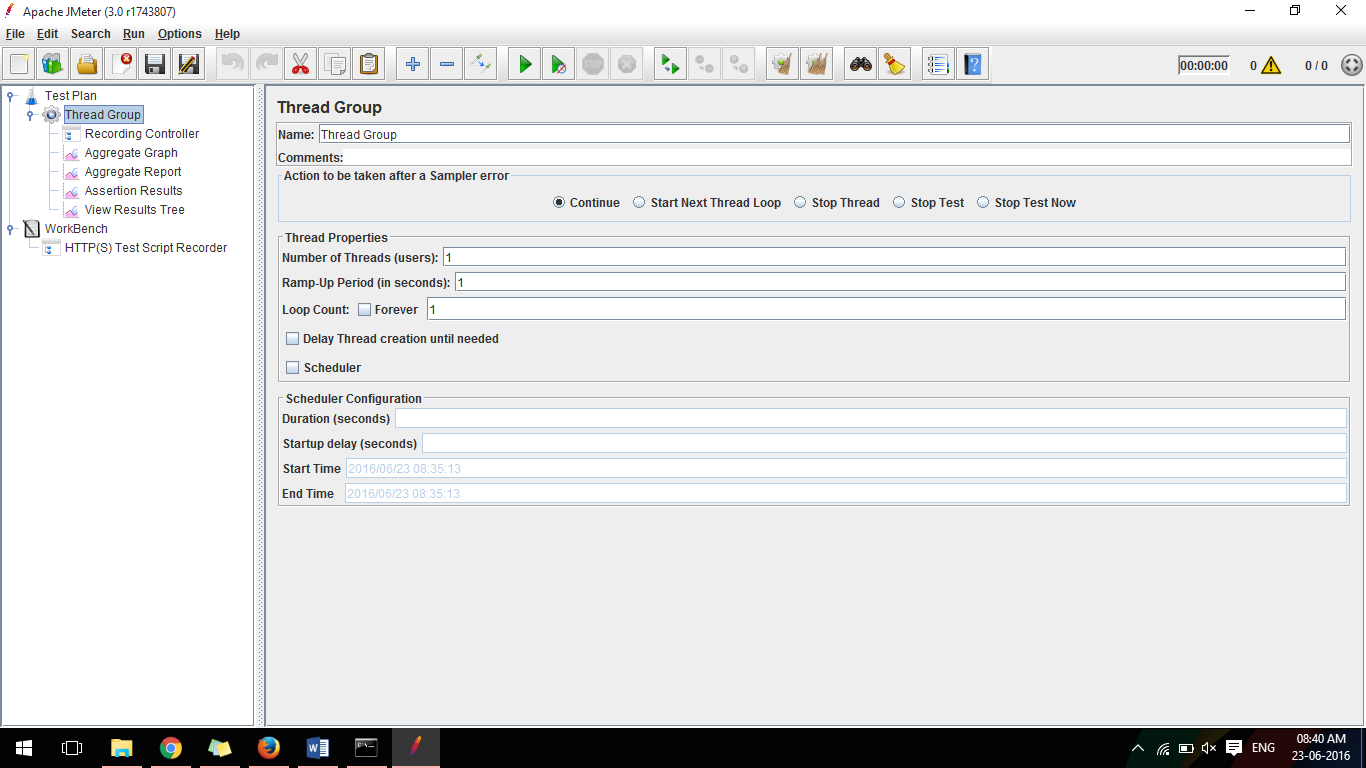
* Download JMeter and extract the zip.
* Open the jmeter.bat file and wait for jmeter to load.



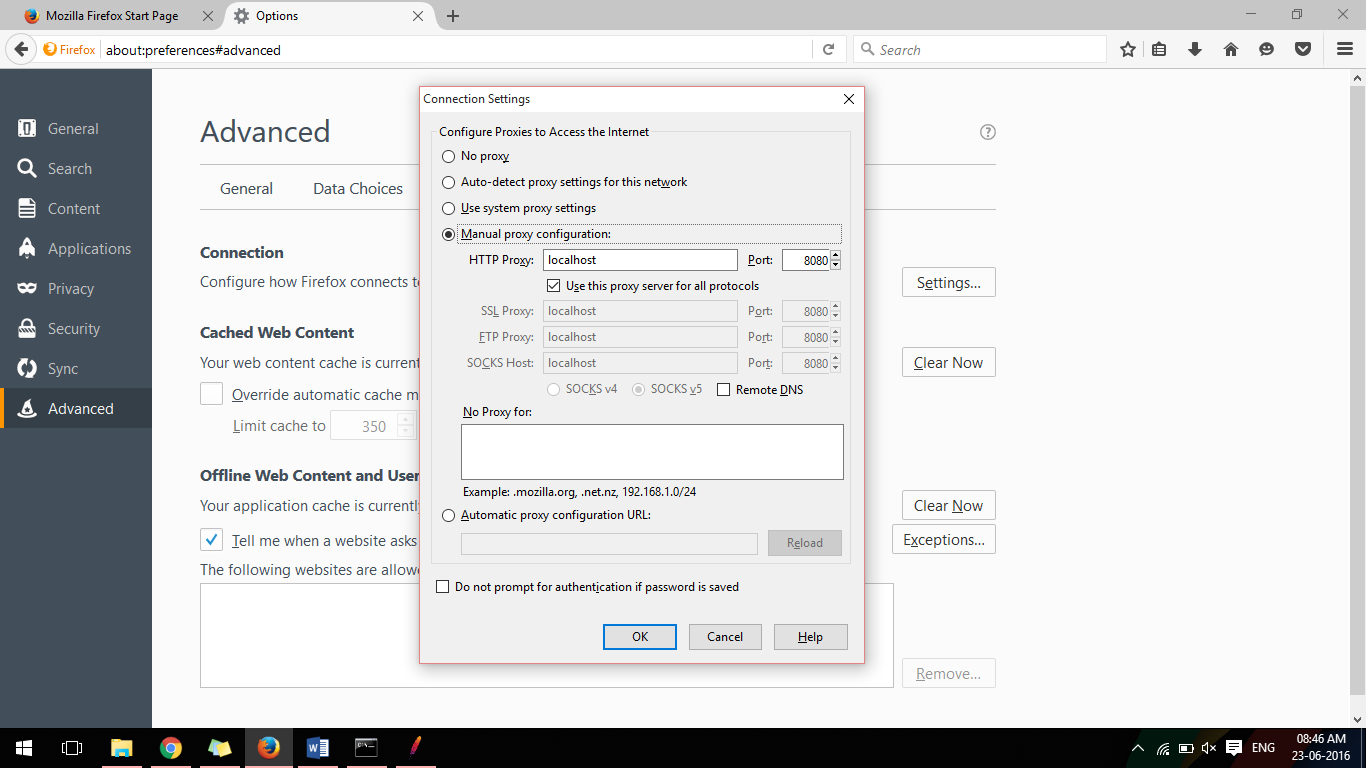
* Right Click on Test Plan go to threads and add a Thread Group



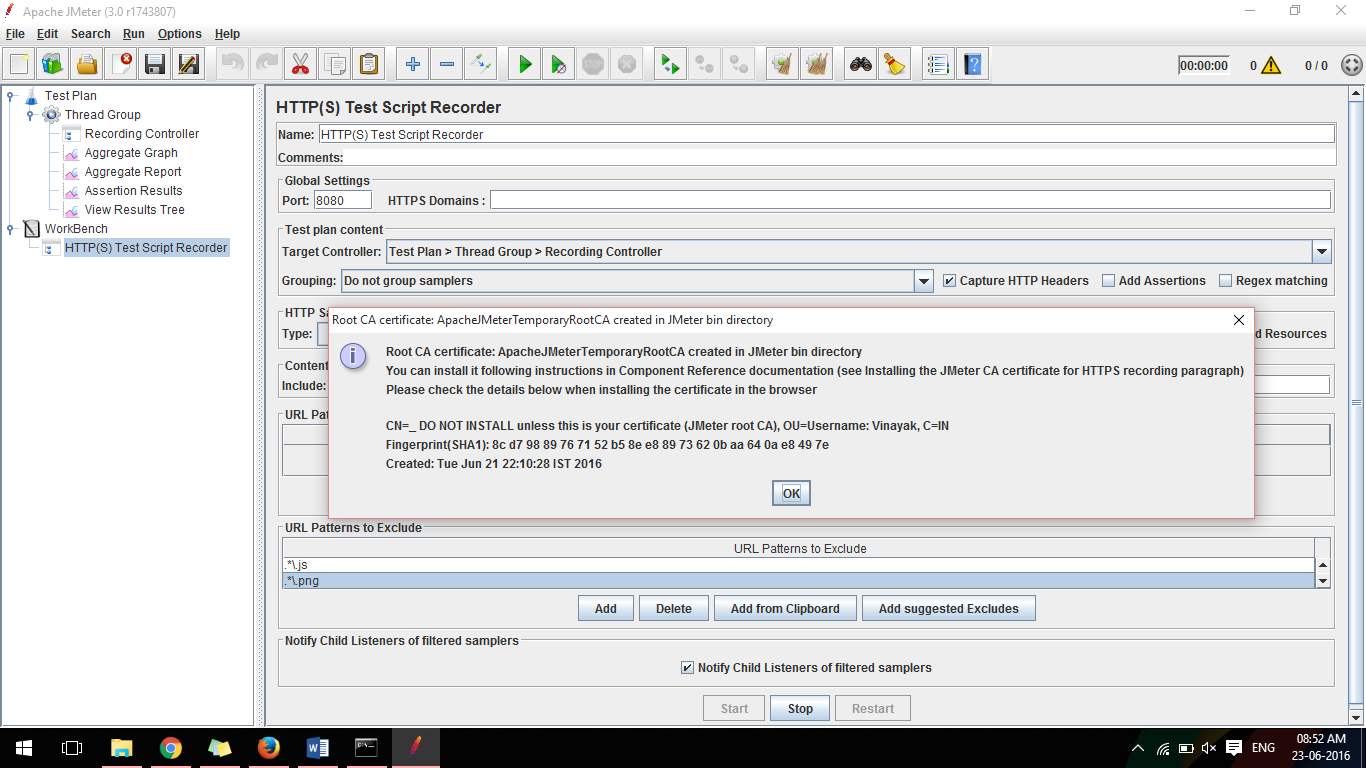
* Right click on Thread Group and go to Logic Controller and add Recording Controller
* Right Click on Thread Group and go to Listener and add Aggregate Graph, Aggregate Report, Assertion Results, and View Results Tree
* Right Click on Workbench and go to Non Test Elements and add HTTP(S) Test Script Recorder, and now your JMeter window wlil look like following



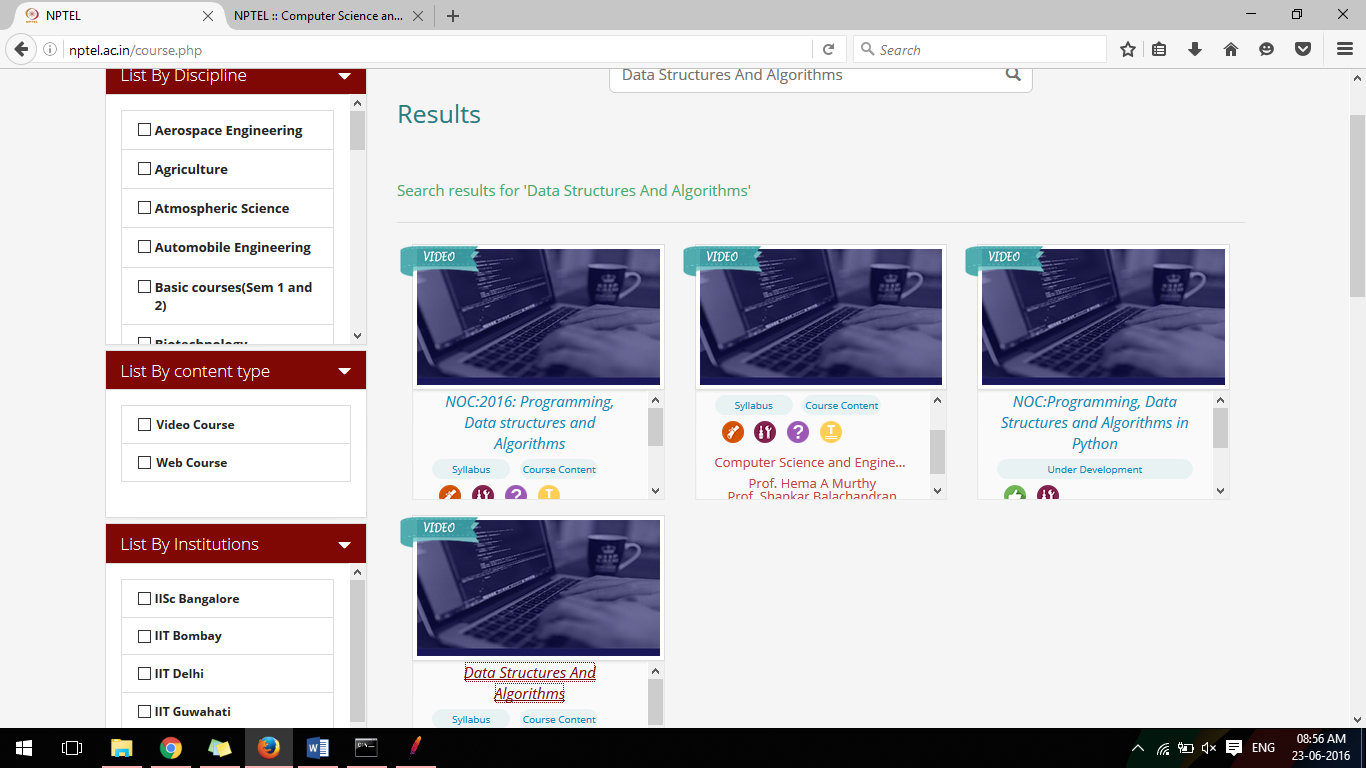
* + - Now we have to configure the proxy in Mozilla Firefox and JMeter so that, JMeter could record the NPTEL website
    - Open Mozilla Firefox -> Go to Options -> Go to Advance -> Go to Network Tab -> Select Connection Setting
    - Select the Manual Proxy option and in HTTP Proxy write : localhost and in Port write : 8080, select use this proxy for all web servers, remove any text written in ‘No Proxy For’ Textbox and click OK. And it will look like as follows



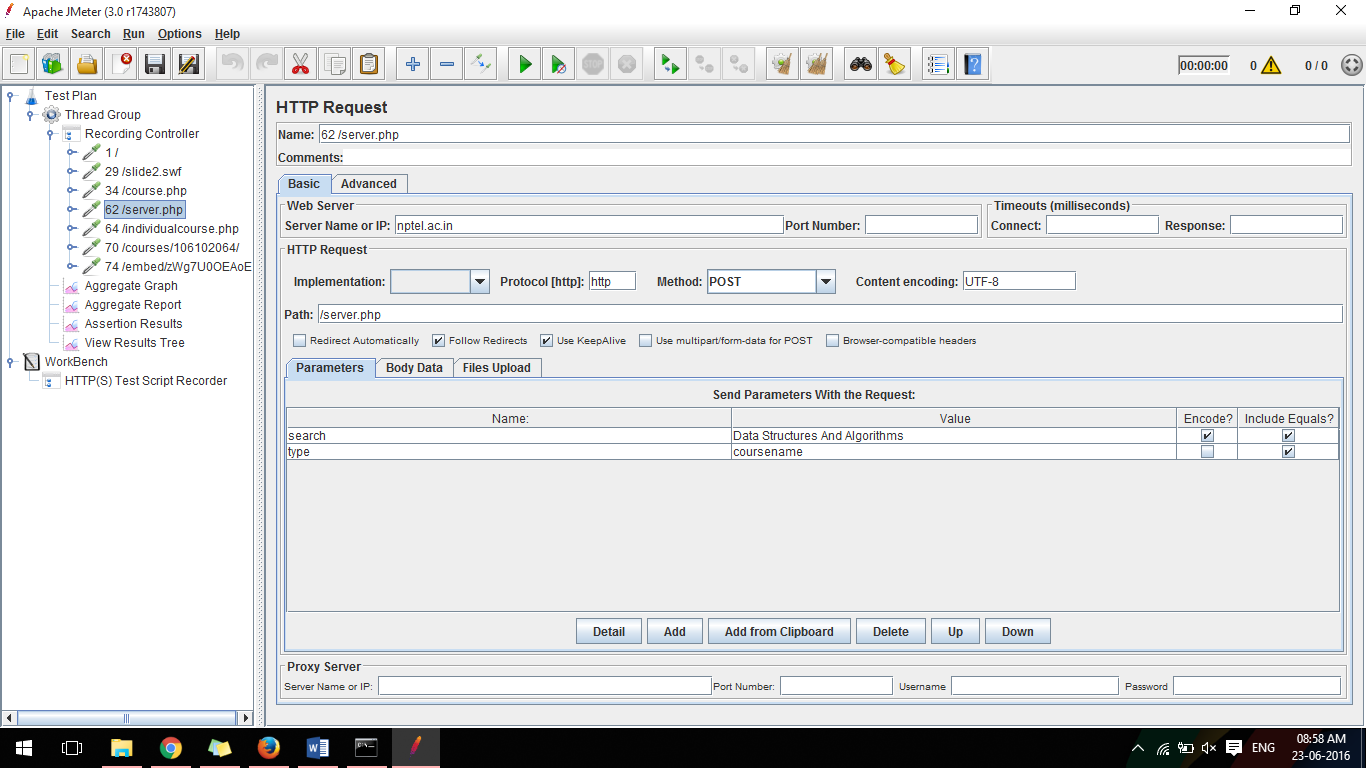
* Go to JMeter and select HTTP(S) Test Script Recorder in Workbench, and then write the Port no. as 8080 and in Target Controller Select Test Plan ->Thread Group->Recording Controller, go to URL patterns Exclude and add following : .\*\.jpg , .\*\.css , .\*\.js , .\*\.png , Click on start button and a small window will open and click ok again, it will look as follows

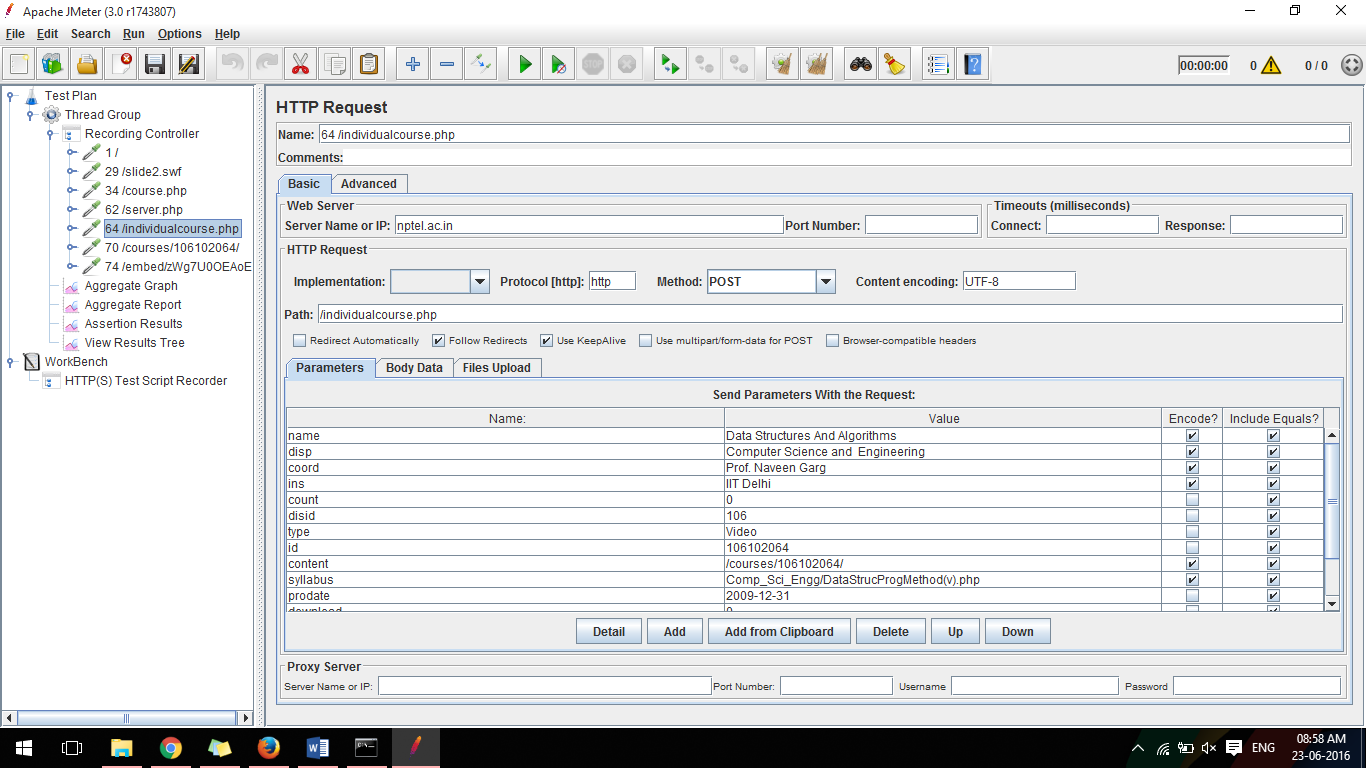


* Now JMeter is working, Now go to Mozilla Firefox and visit nptel.ac.in now click on courses and search Data Structures and Algorithms, click on any one of the search results

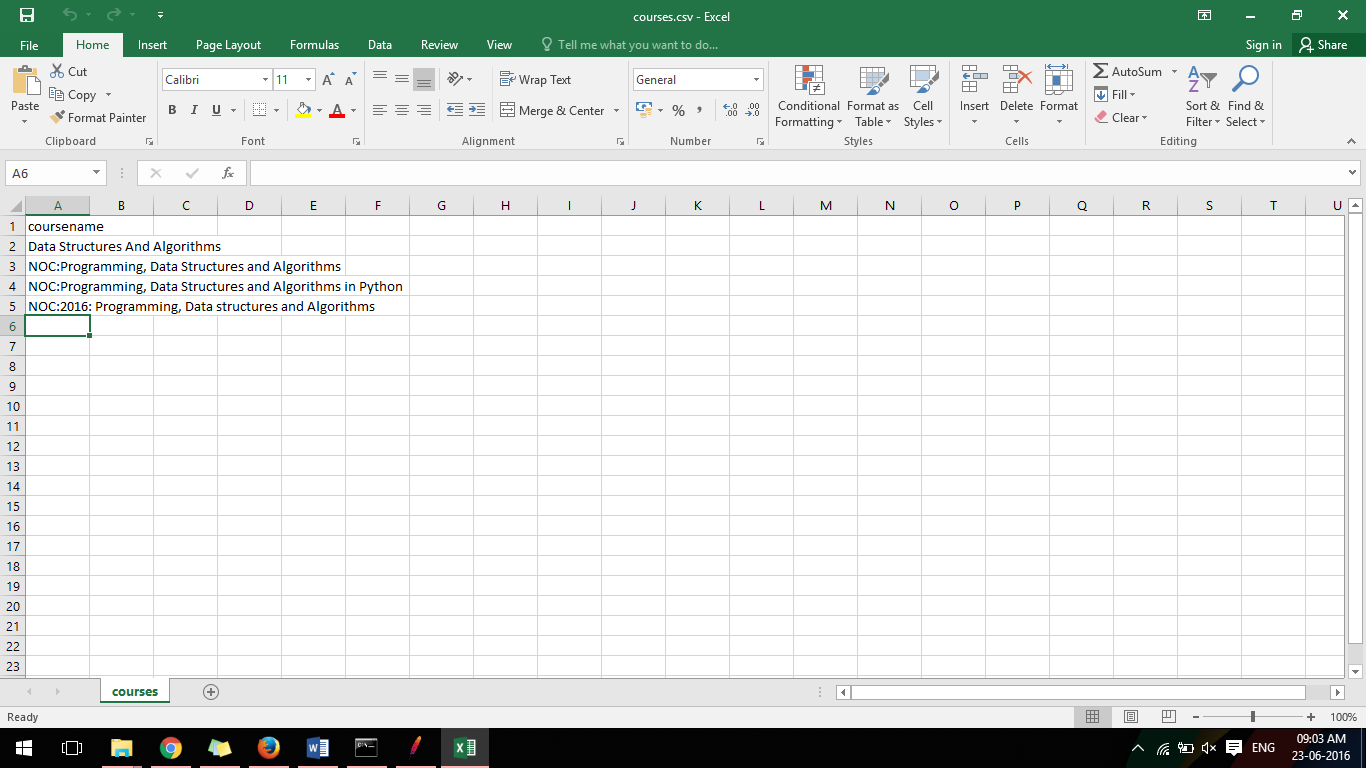


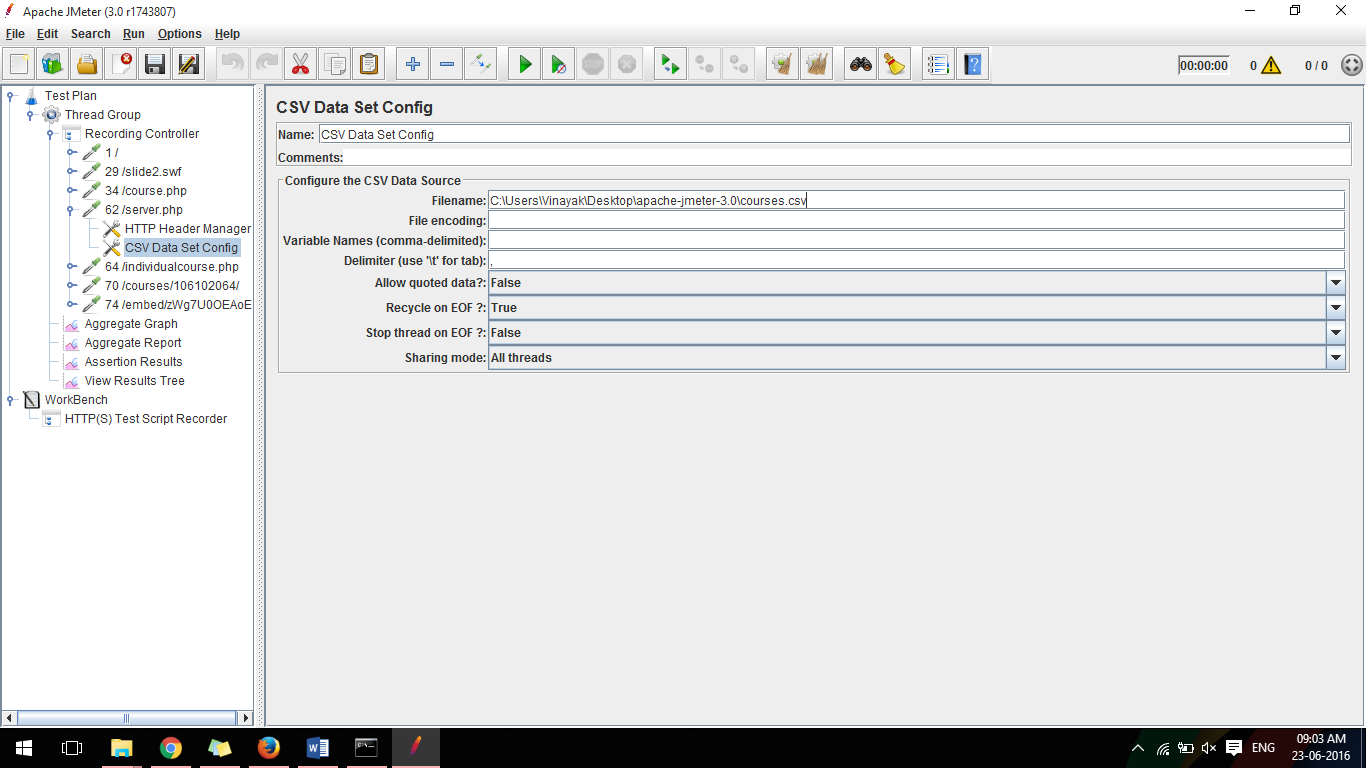
* Now go to Jmeter and go to Recording Controller in Thread Group, delete all the server.php files except the last one, delete all the individual course except the one whom you clicked and opened in Mozilla, it will look as follows



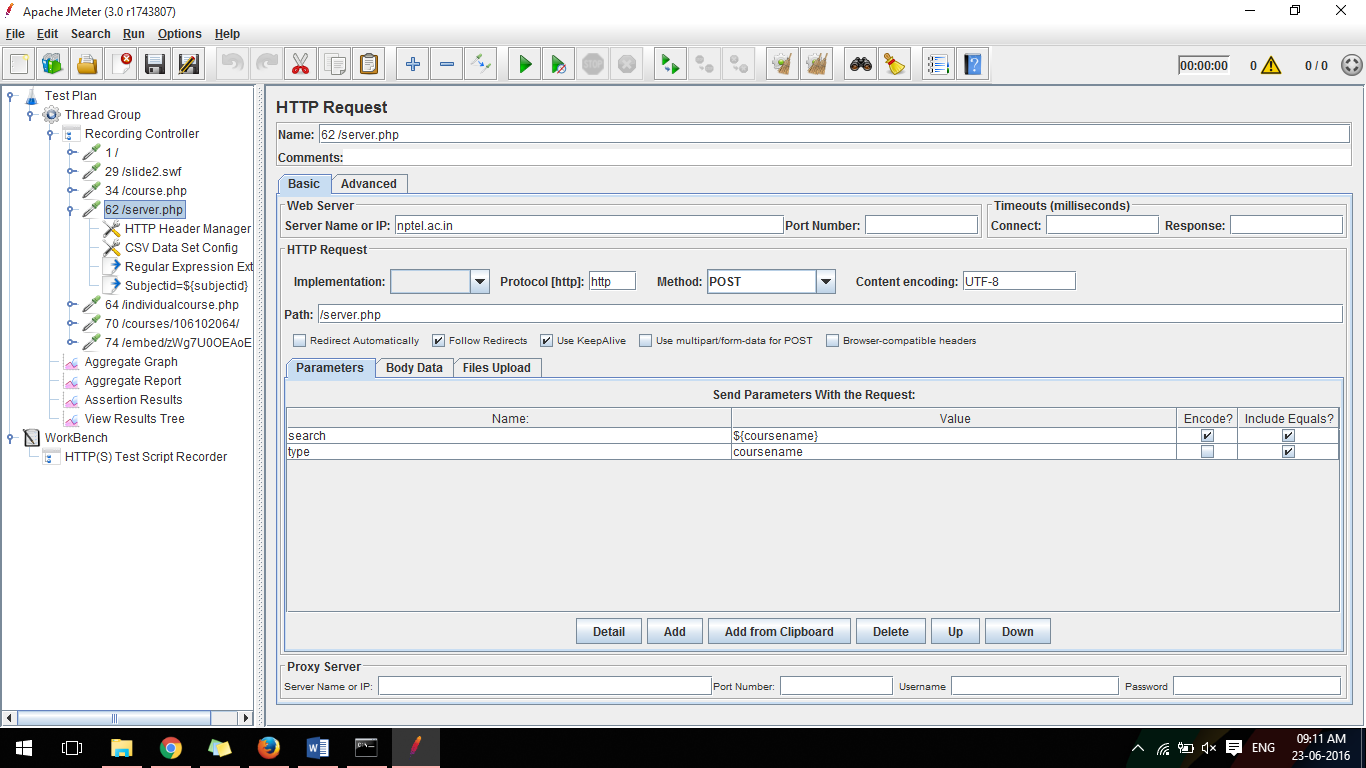


* Now right click on server.php and add config element CSV Data Set Config, in filename add the path of the CSV file having coursename of all the courses you want to search and record in JMeter, the CSV file and JMeter will look as follows

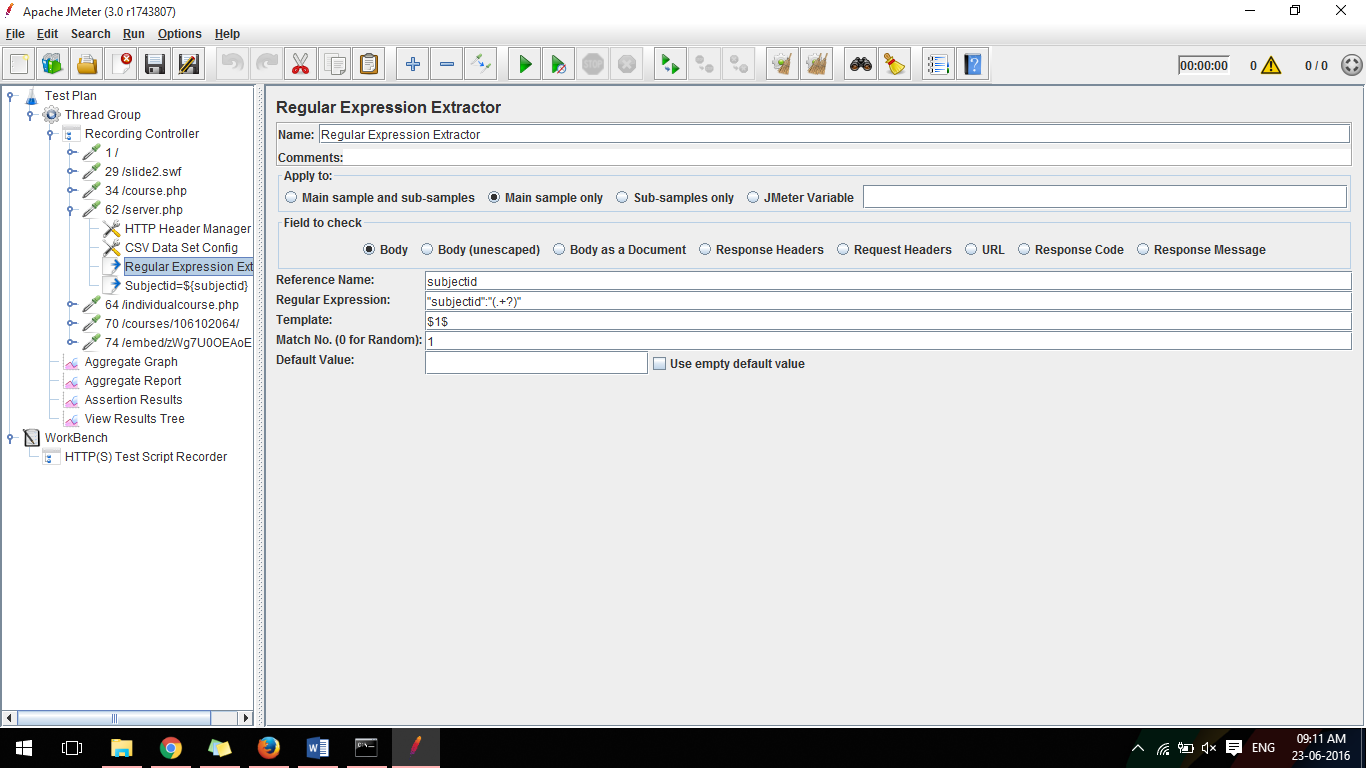




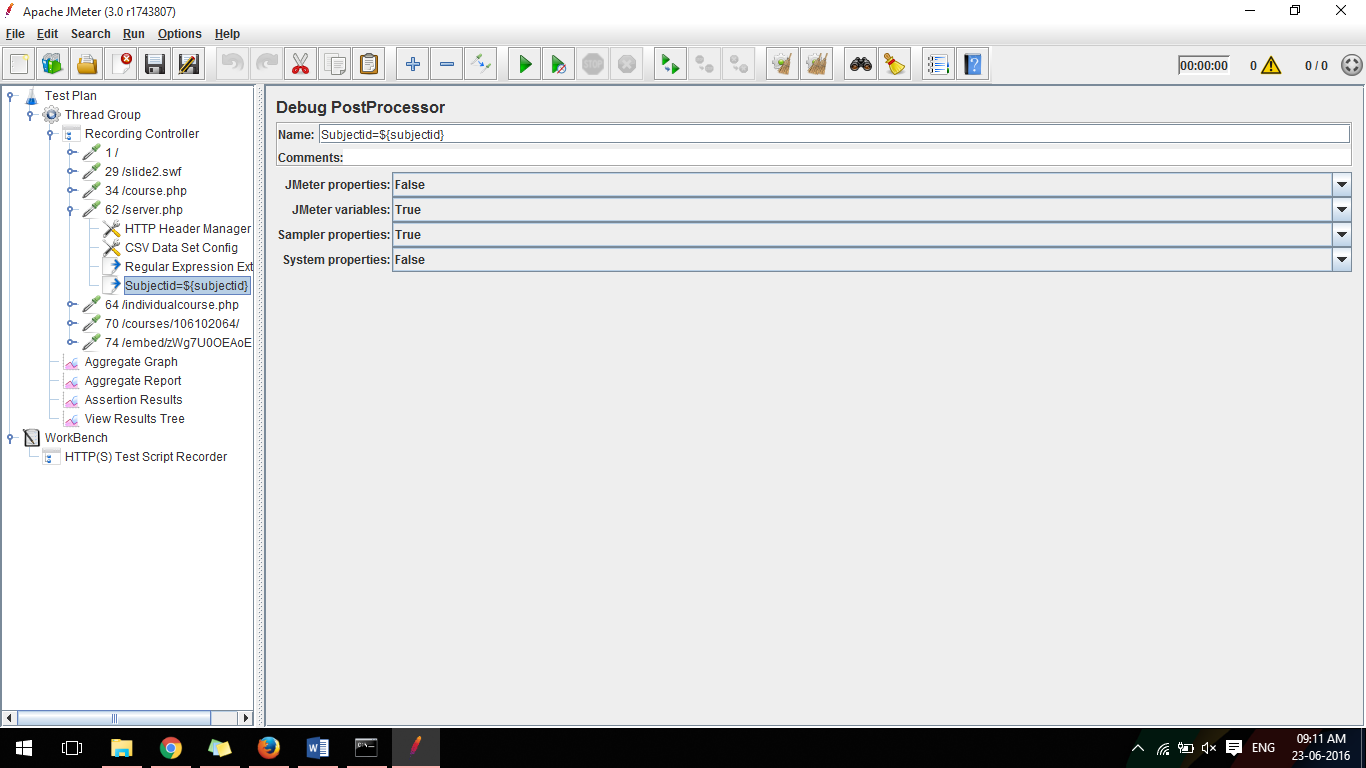
* Now right click on server.php and add postprocessor and select Regular Expression Extractor, and Debug PostProcessor
* Click on server.php and go to search and change its value with ${coursename}



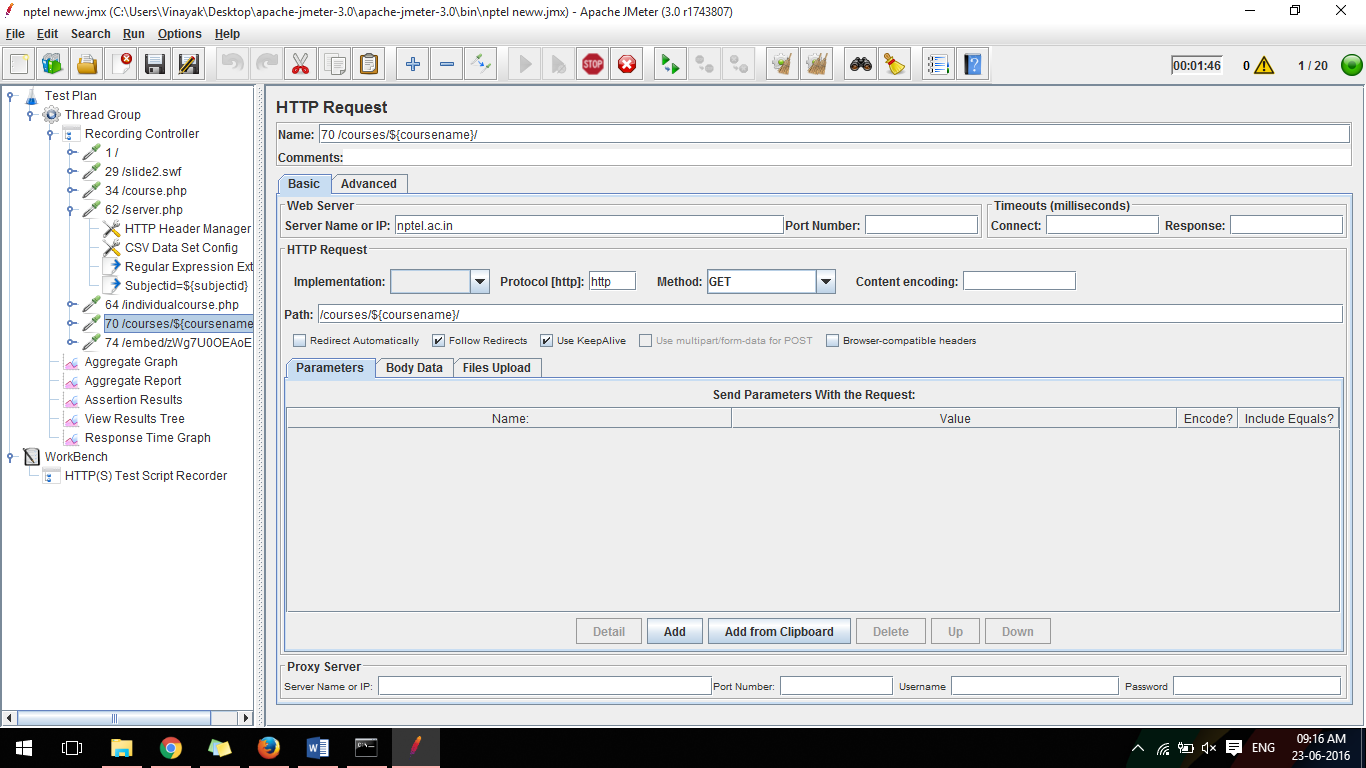
* Go to Regular Expression Extractor and in reference name write : subjected, in regular expression write : “subjectid”:”(.+?)”, in template write : $1$, in Match no. write : 1



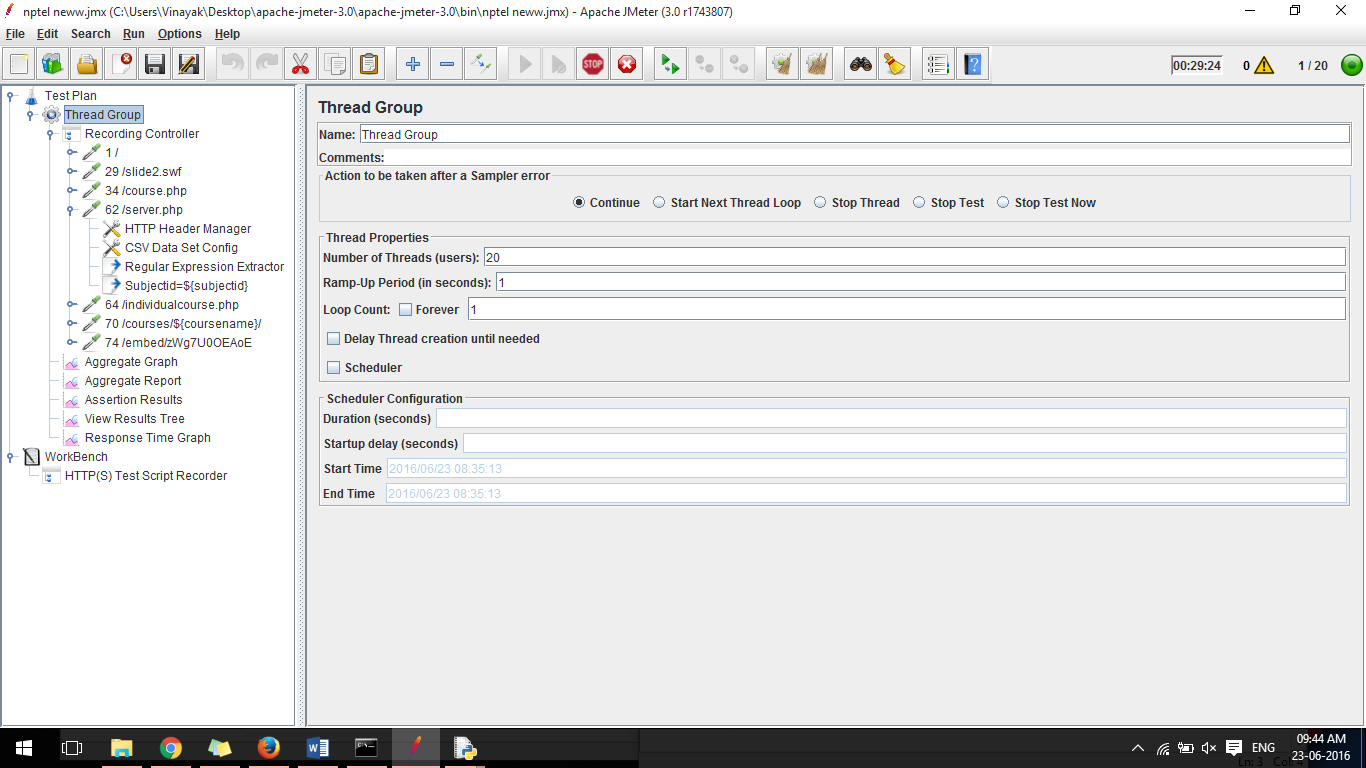
* Go to Debug Post Processor and change it’s name as ‘Subjectid=${subjectid}’



* Go to Courses and replace the course no. with variable name ${coursename}, And same way change the course no. in path with ${coursename}



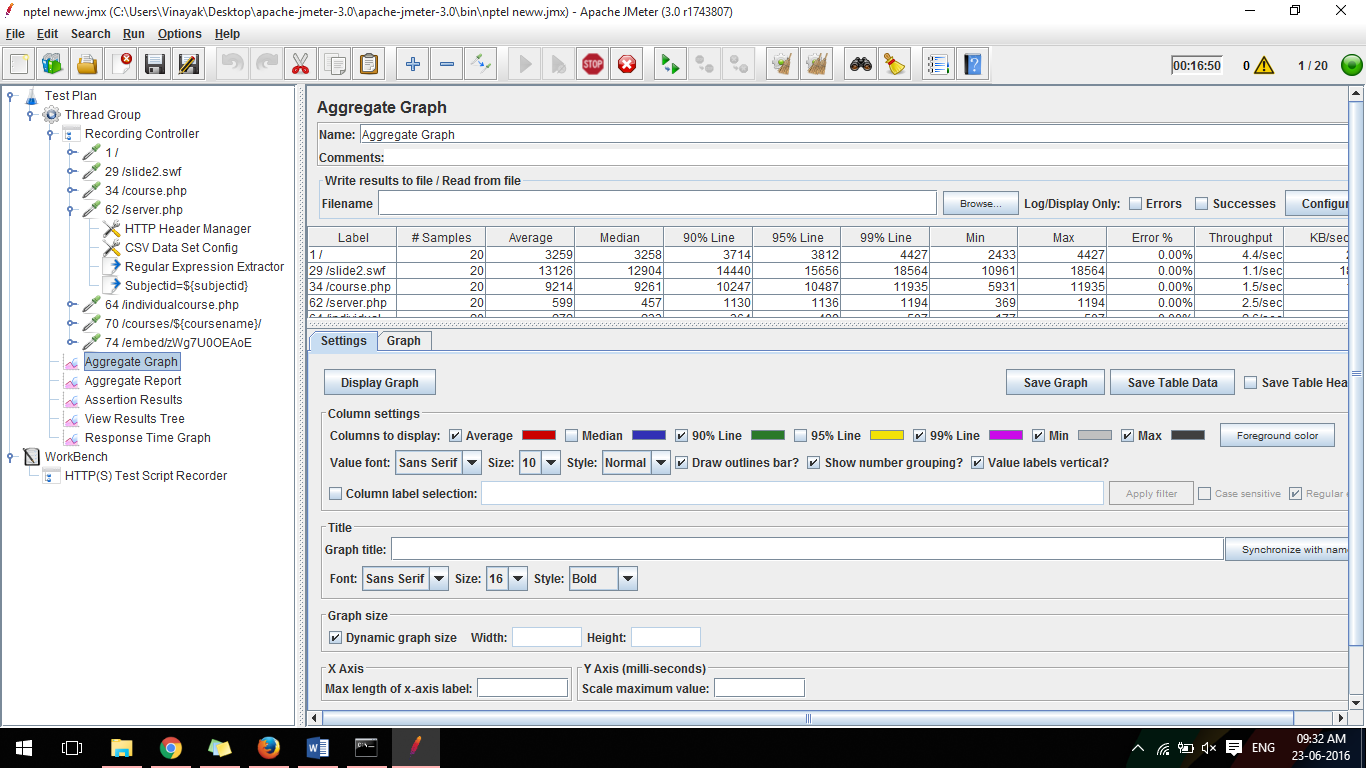
* Right click on Thread Group and add listeners Response Time Graph.
* Go to Thread group and change Number of Threads to 20

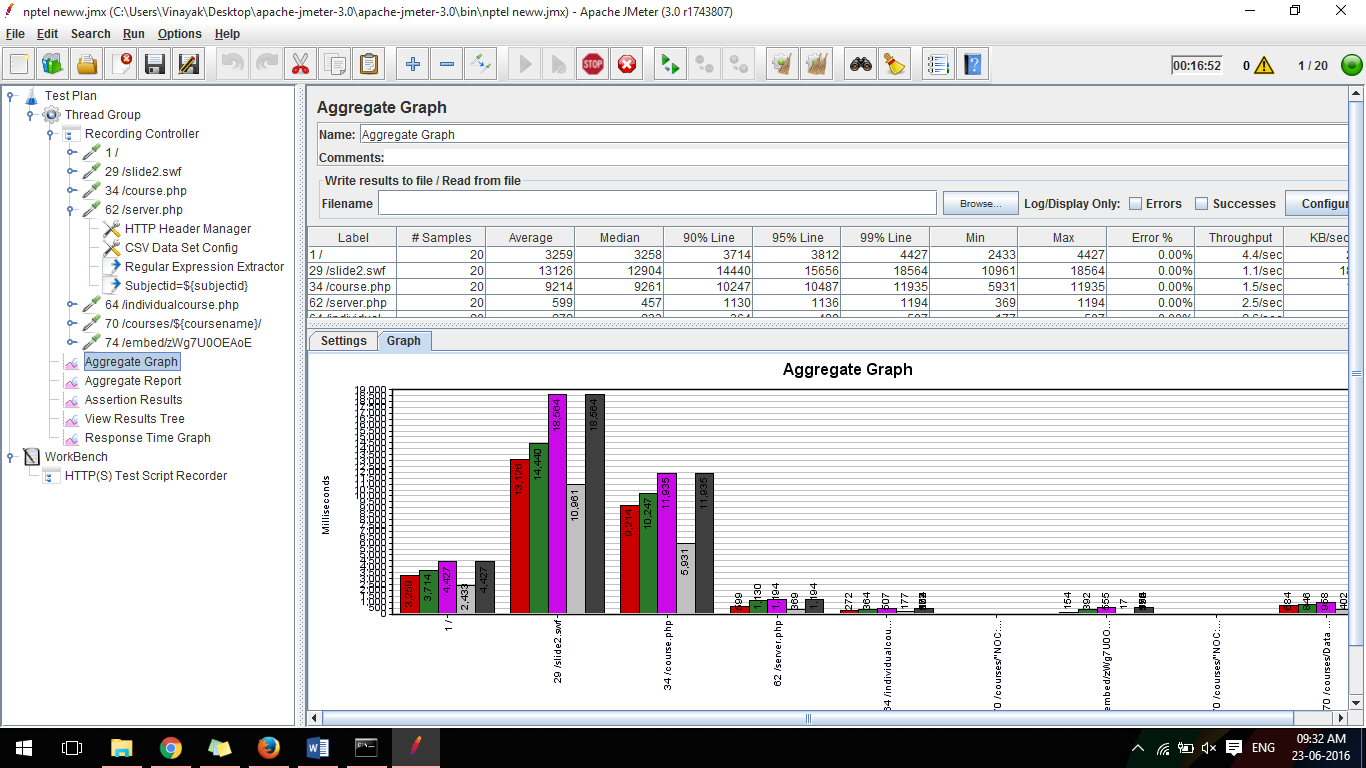


* Now Run the Script using the Green Button at top, and now we can see different results using the various listeners we added in the Thread Group,

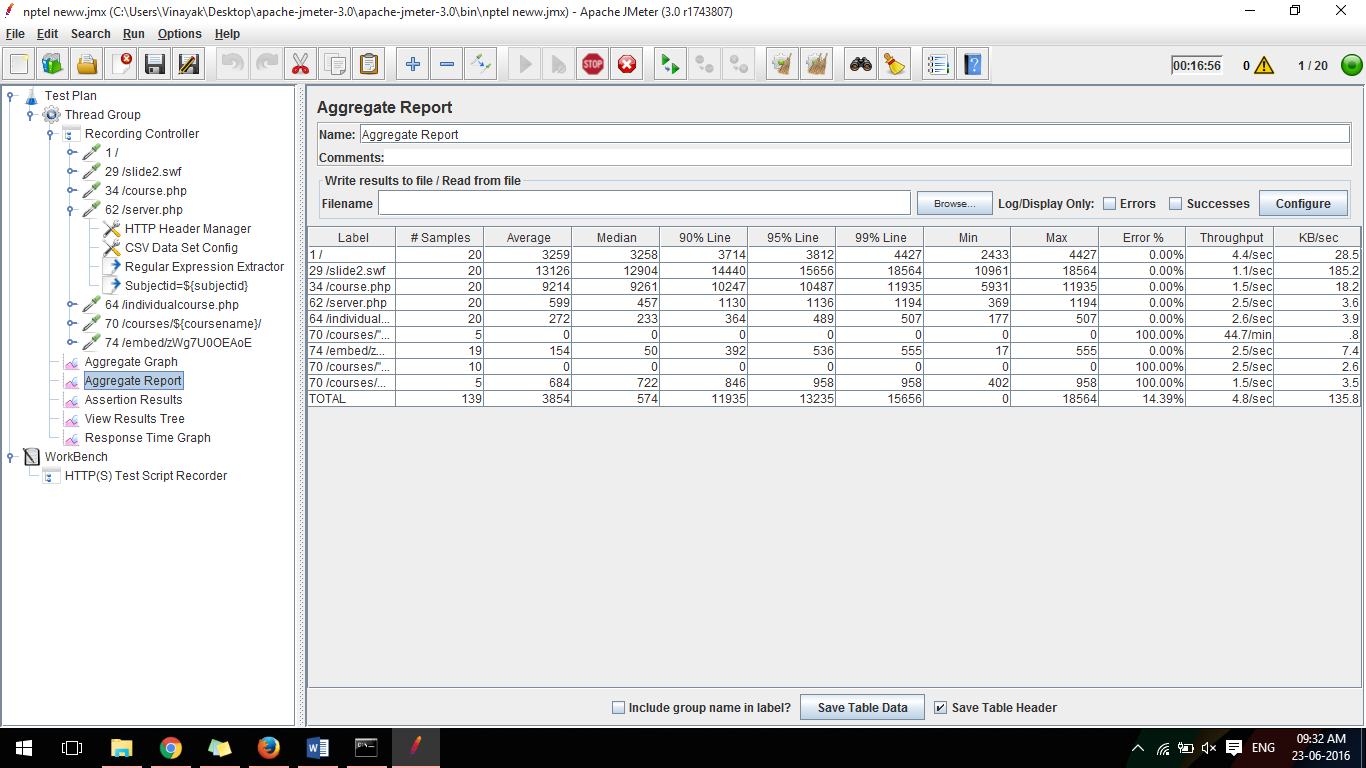
Go to aggregate Graph select 90%line, 99& line, Min, Max, following are the screenshots of various results

**Aggregate Graph:-**

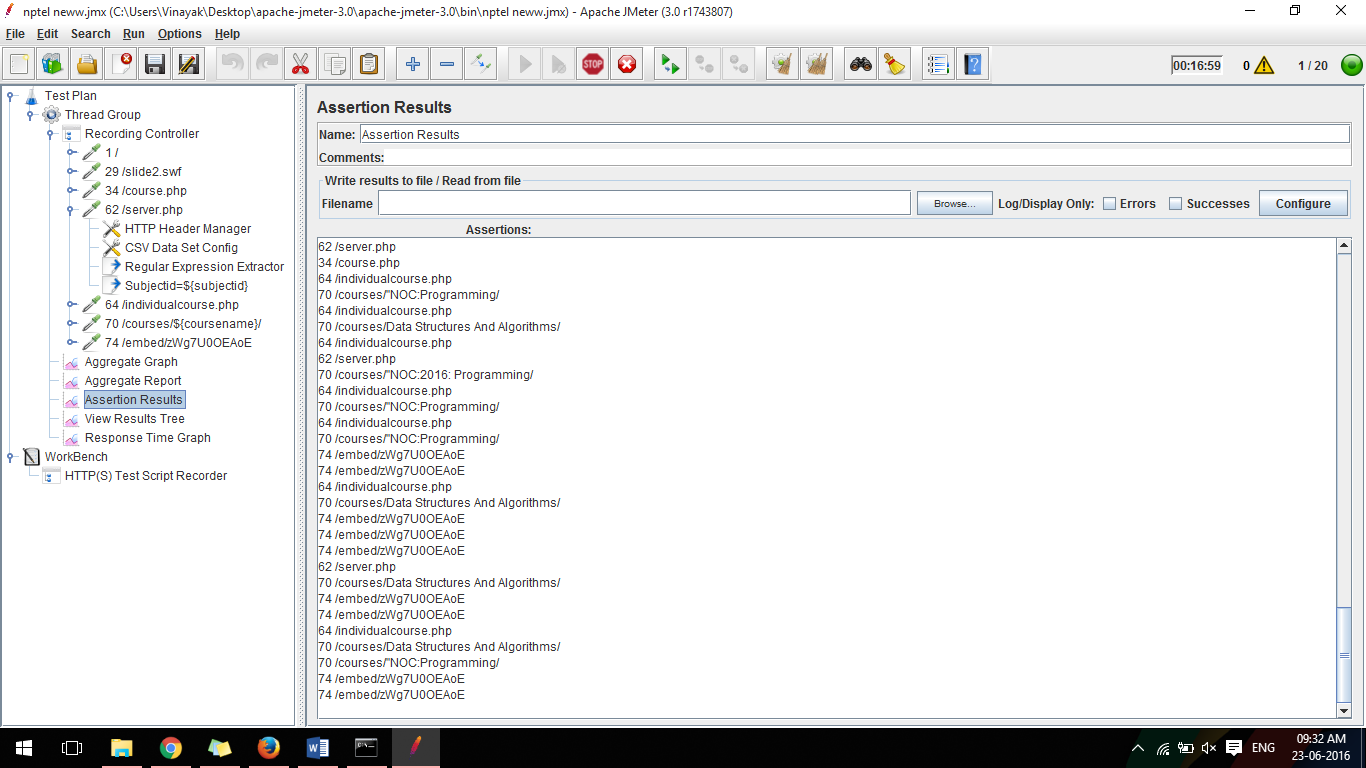




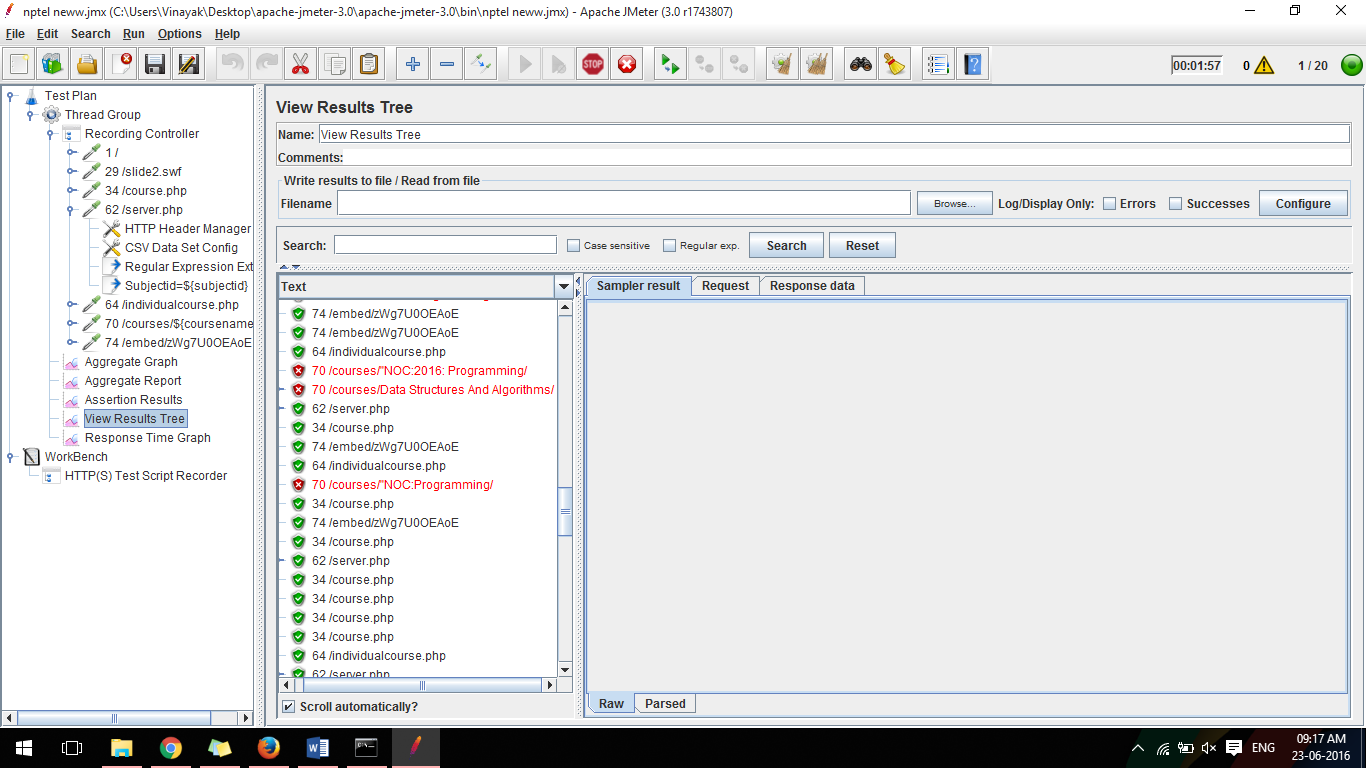
**Aggregate Report:-**

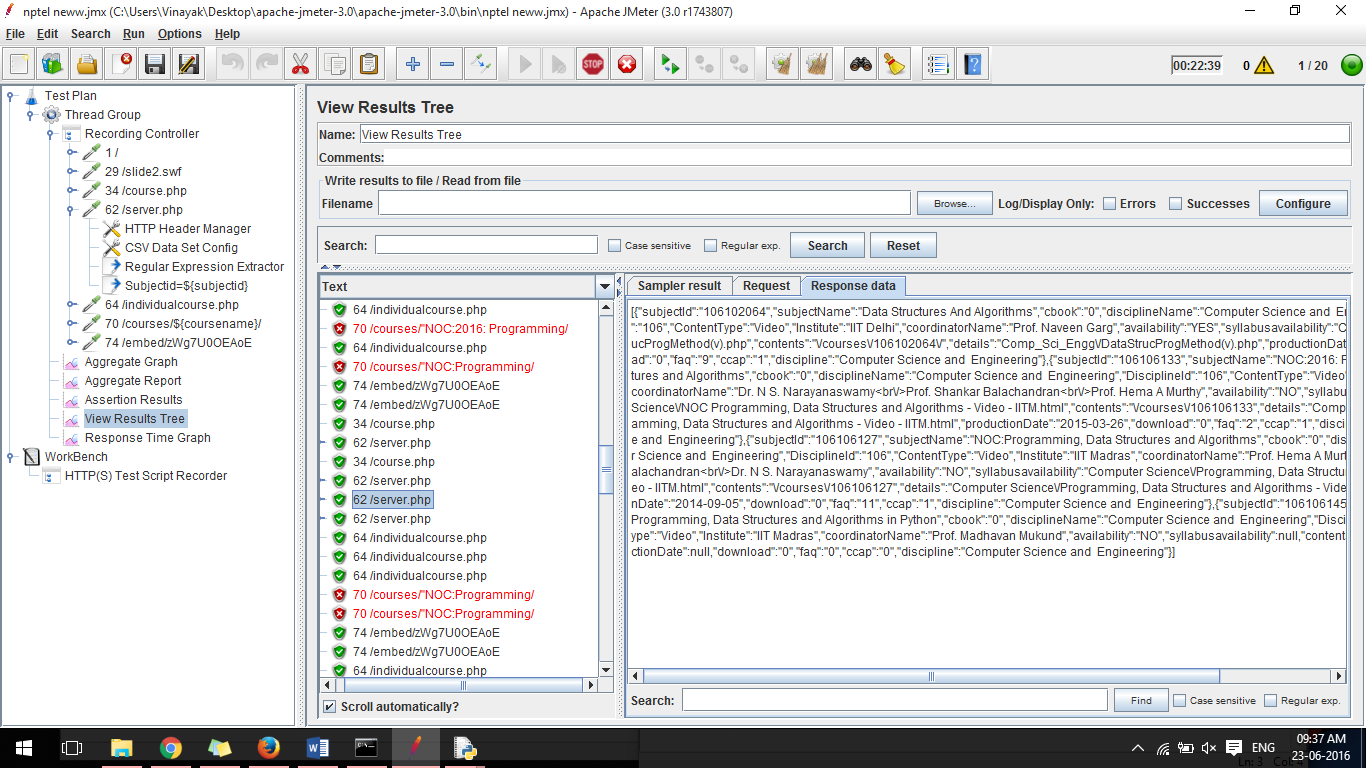


**Assertion Results**

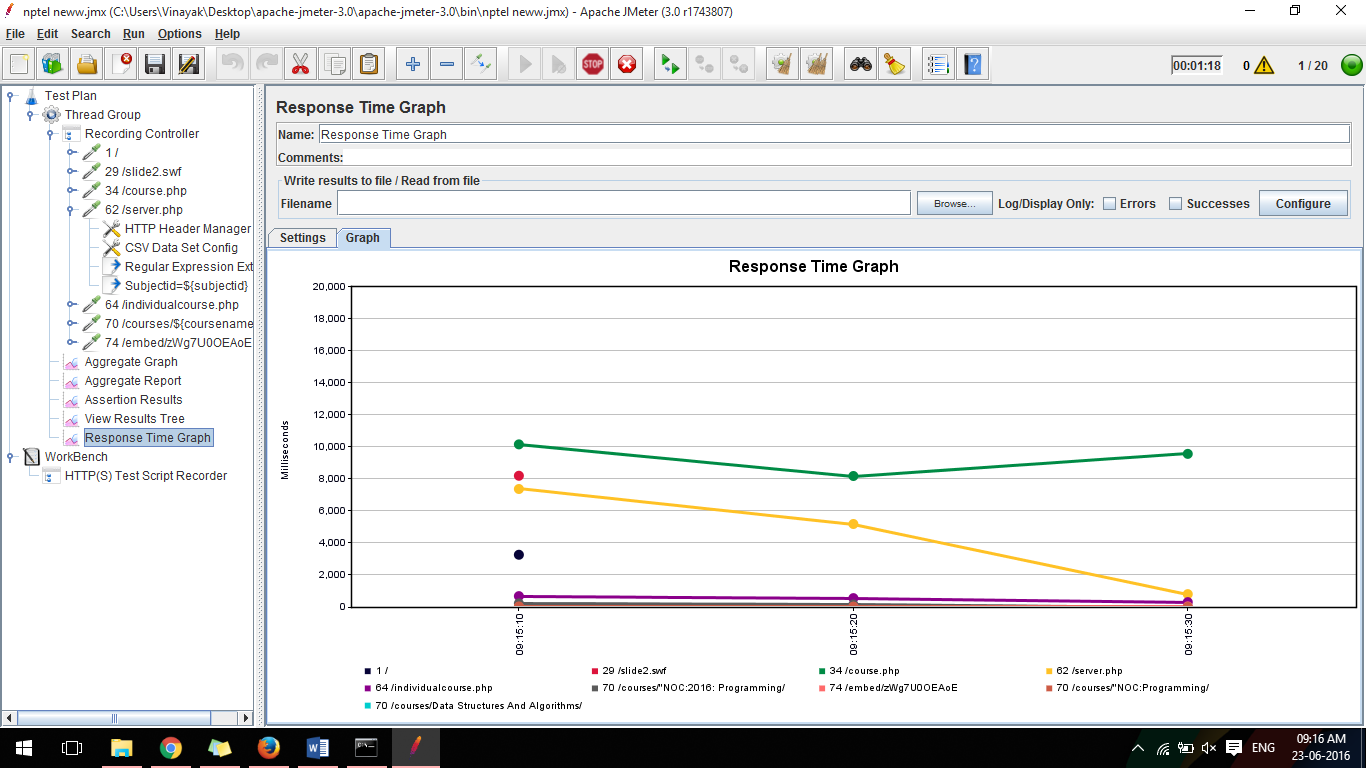


**View Results Tree:-**





**Response Time Graph:-**



Now we have recorded all the results of nptel.ac.in, now we can analyse the performance and various other aspects

# Conclusion:

# For the success of Software system Performance testing plays a key role software testing life cycle. It makes certain the customer satisfaction & elements the risk of product failure. So it very much important to carry out the Performance testing before goes live in the marke

PYTHON

**1.Write a program to print string hello.**

print('my program')

**output:**

my program

**2. Write a program to print string my.**

print('my')

**Output:**

my

**3.Write a program to check whether no is even or odd.**

num=int(input('enter a no.:'))

if num%2==0:

print('no is even')

else:

print('no is odd')

**Output:**

enter a no.:44

no is even

**4.Write a program to check whther n1 is greater or n2.**

n1=input('enter a n1:')

n2=input('enter a n2:')

if n1>n2:

print('n1 is greater ')

else:

print('n2 is greater')

**Output:**

enter a n1:1

enter a n2:8

n2 is greater

**5.Write a program to check whther no is positive or not.**

num=int(input('enter a no:'))

if num>0:

print('no is +ve')

else:

print('no is -ve')

**Output:**

enter a no:48

no is +ve

**6. Write a program to check which number is greatr amongst three numbers using nested else-if(elif)**

n1=input('enter a n1:')

n2=input('enter a n2:')

n3=input('enter a n3:')

if n1>n2:

if n1>n3:

print('n1 is greatest')

else:

print('n2 is greatest')

else:

if n2>n3:

print('n2 is greatest')

else:

print('n3 is greatest')

**Output:**

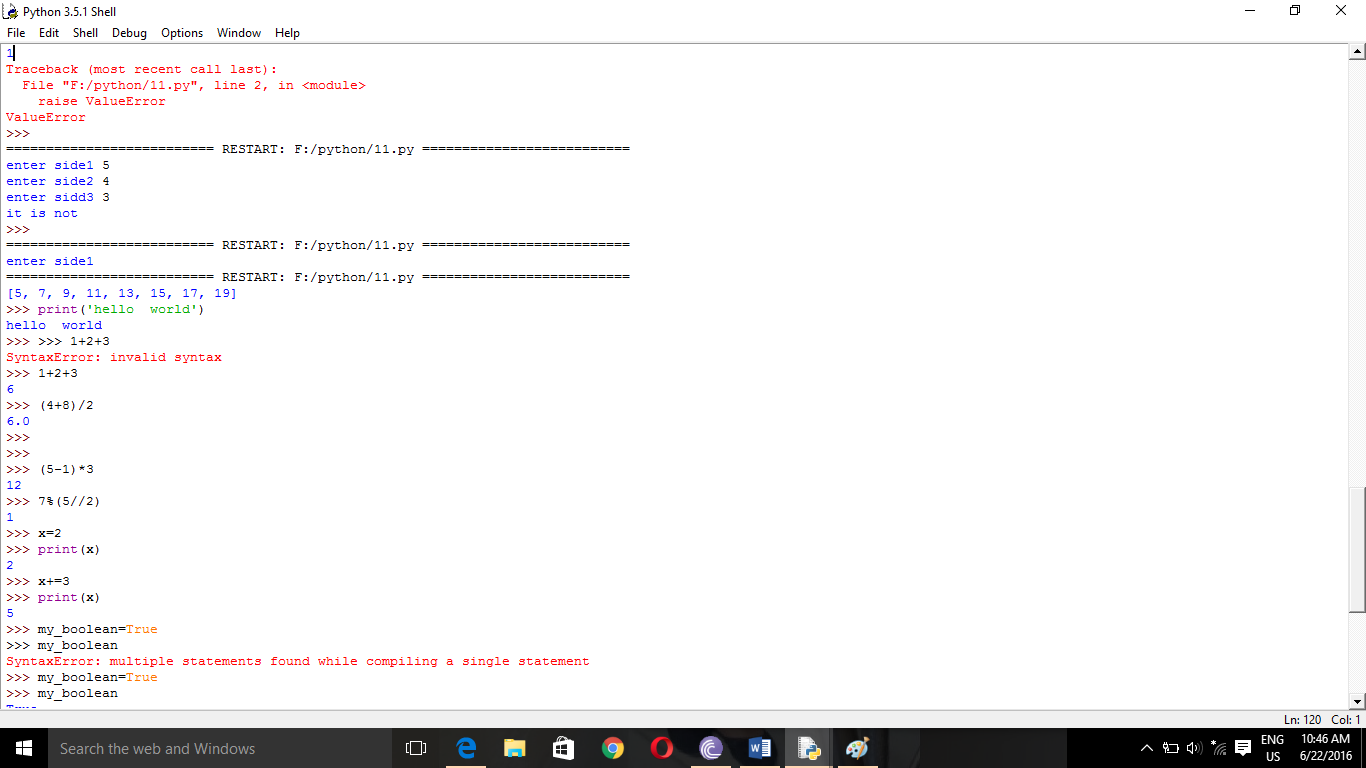
enter a n1:2

enter a n2:8

enter a n3:3

n2 is greatest

**7. ADD,MULTIPLY,DIVIDE,SUBTRACT**



**8.ELIF**

num=int(input('enter num:'))

if num==5:

print("num is 5")

elif num==11:

print("num is 11")

elif num==7:

print("num is 7")

else:

print("num isnt 5,11 or 7")

**Output:**

enter num:12

num isnt 5,11 or 7

>>>

enter num:11

num is 11

**9.**

num=int(input('enter no.:'))

if num>5:

print("bigger than 5")

if num<=23:

print("bw 5 and 46")

**Output:**

enter no.:24

bigger than 5

enter no.:19

bigger than 5

bw 5 and 46

**10.**

n1=int(input('enter a n1:'))

n2=int(input('enter a n2:'))

n3=int(input('enter a n3:'))

if (n1\*n1)==(n2\*n2+n3\*n3):

print('traiangle is rt angled')

elif(n2\*n2)==(n3\*n3+n1\*n1):

print('traiangle is rt angled')

elif(n3\*n3)==(n2\*n2+n1\*n1):

print('traiangle is rt angled')

else:

print('not a rt angled')

**Output:**

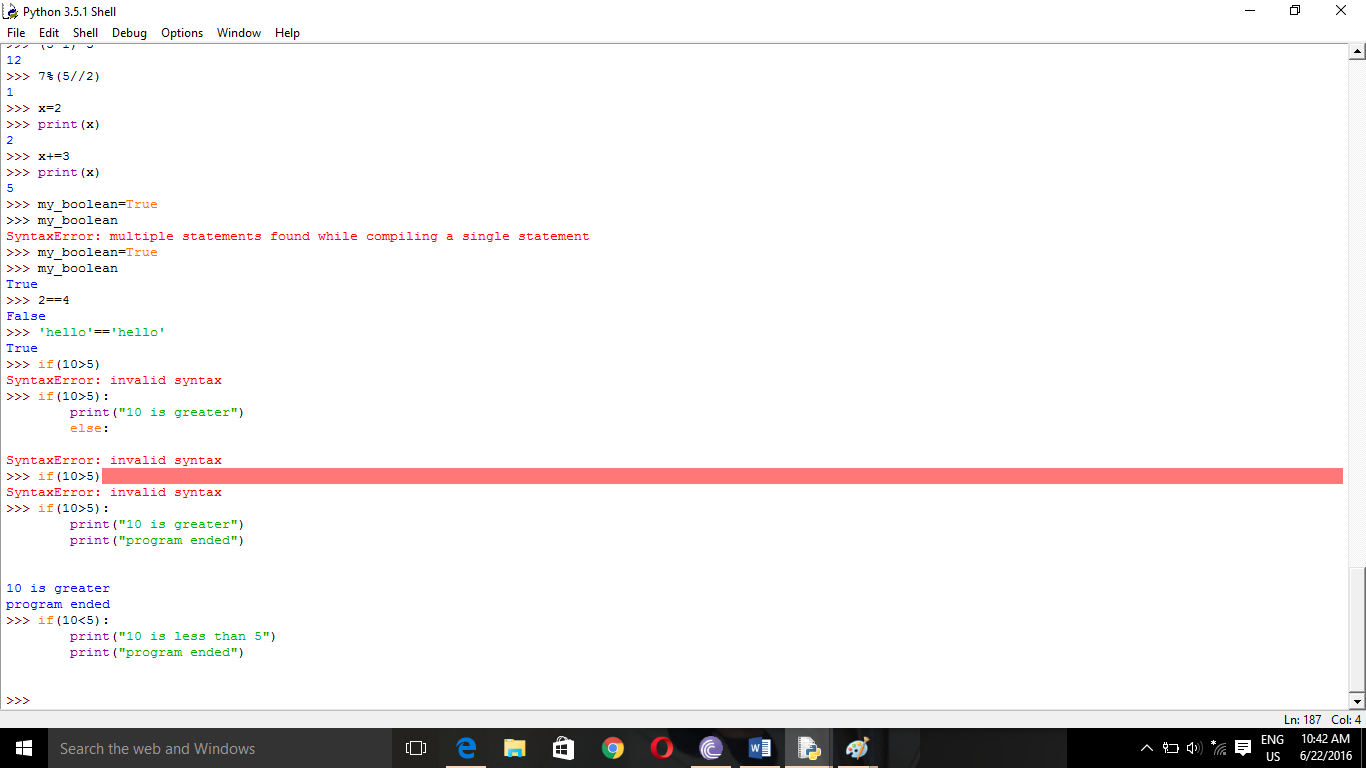
enter a n1:4

enter a n2:3

enter a n3:5

traiangle is rt angled

**11. BOOLEAN**



**12.WHILE LOOP**

i=1

while i<=5:

print(i)

i=i+1

print('Finished')

**Output:**

1

2

3

4

5

Finished

**13.**

i=0

while 1==1:

print(i)

i=i+1

if i>=3:

print('breaking ')

break

print('finished')

**Output:**

0

1

2

breaking

finished

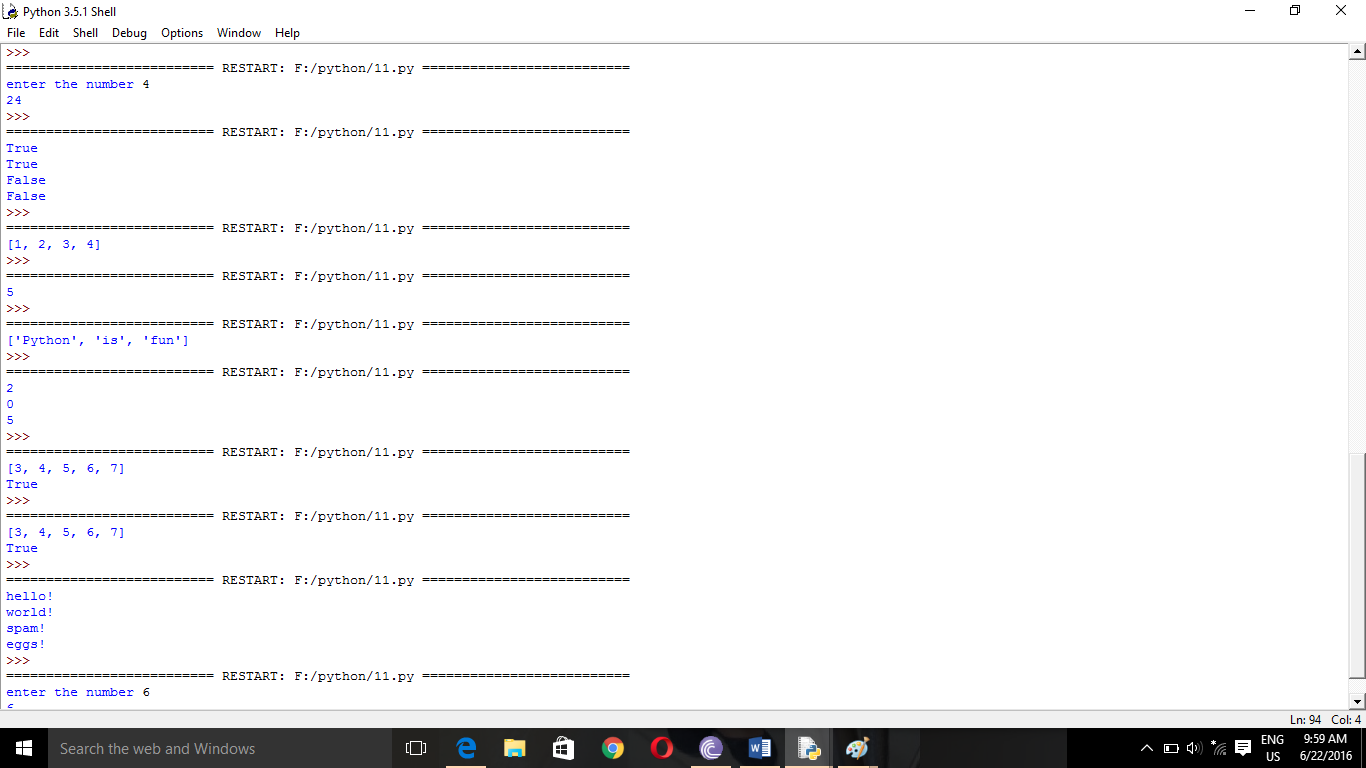
**14. APPEND**

nums=[1,2,3]

nums.append(4)

print(nums)

OUTPUT:-

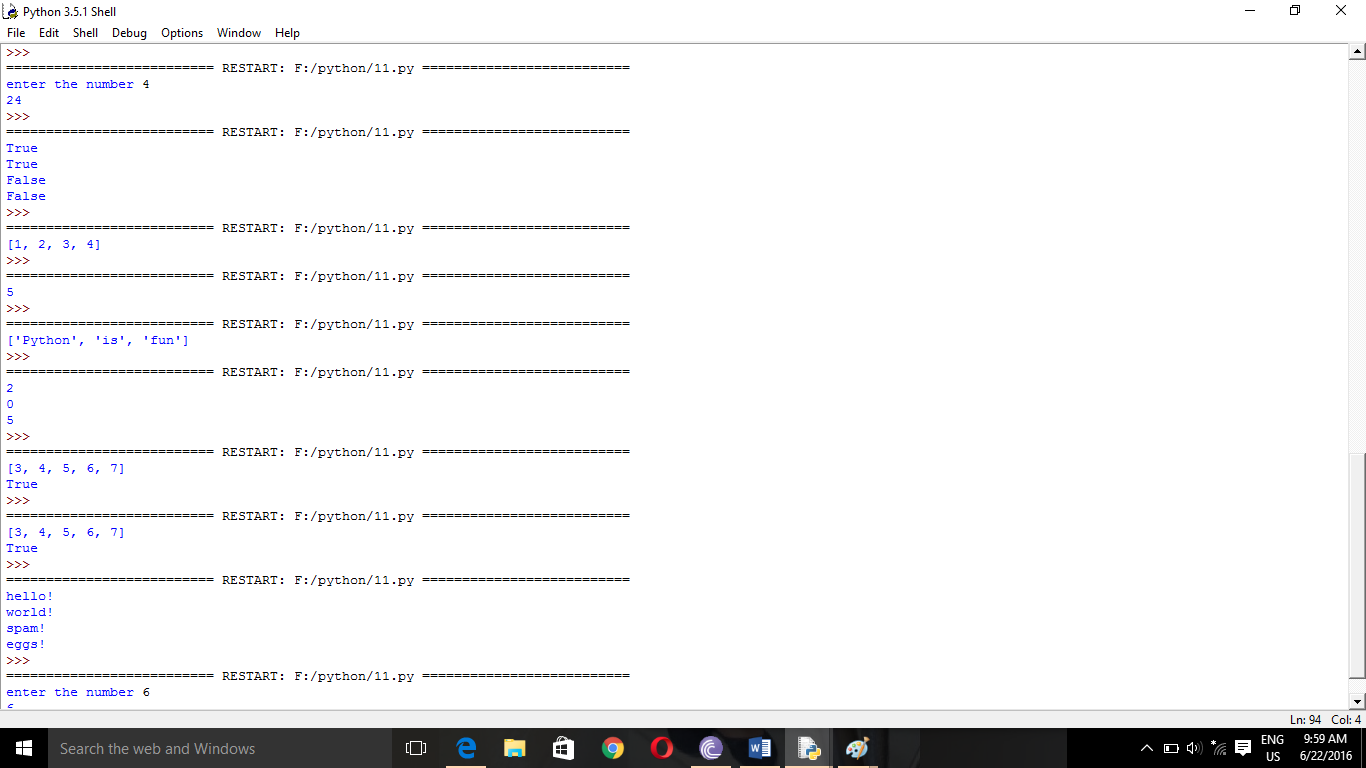


**LENGTH**

nums=[1,3,5,2,4]

print(len(nums))

OUTPUT:-



**15.**

n=int(input('enter n:'))

i=1

while i<=n:

print('hello')

i+=1

**Output:**

enter n:4

hello

hello

hello

hello

**CONTINUE STATEMENT**

i=0

while True:

i+=1

if i==2:

print('skipping 2')

continue

if i==5:

print('breaking')

break

print(i)

print('finished')

**Output:**

1

skipping 2

3

4

breaking

finished

**16.BREAK STATEMENT**

i=5

while True:

print(i)

i-=1

if i<=2:

break

**Ouput:**

5

4

3

**17.Write a program to check whether a triangle is a right triangle or not.**

a=int(input('enter a a.:'))

b=int(input('enter a b.:'))

c=int(input('enter a c:'))

if (a\*a)==(b\*b)+(c\*c) or (b\*b)==(c\*c)+(a\*a) or (c\*c)==(a\*a)+(b\*b):

print('rt triangle')

else:

print('not a rt angled')

**Output:**

enter a a.:4

enter a b.:3

enter a c:5

rt triangle

**18. Write a program to print factorial of number.**

j=int(input('enter a no'))

i=1

while(j>0):

i=i\*j

j=j-1

print('fact is: \n' +str(i))

**Output:**

enter a no5

fact is:

120

**EXCEPTION HANDLING**

try:

num1=7

num2=0

print(num1/num2)

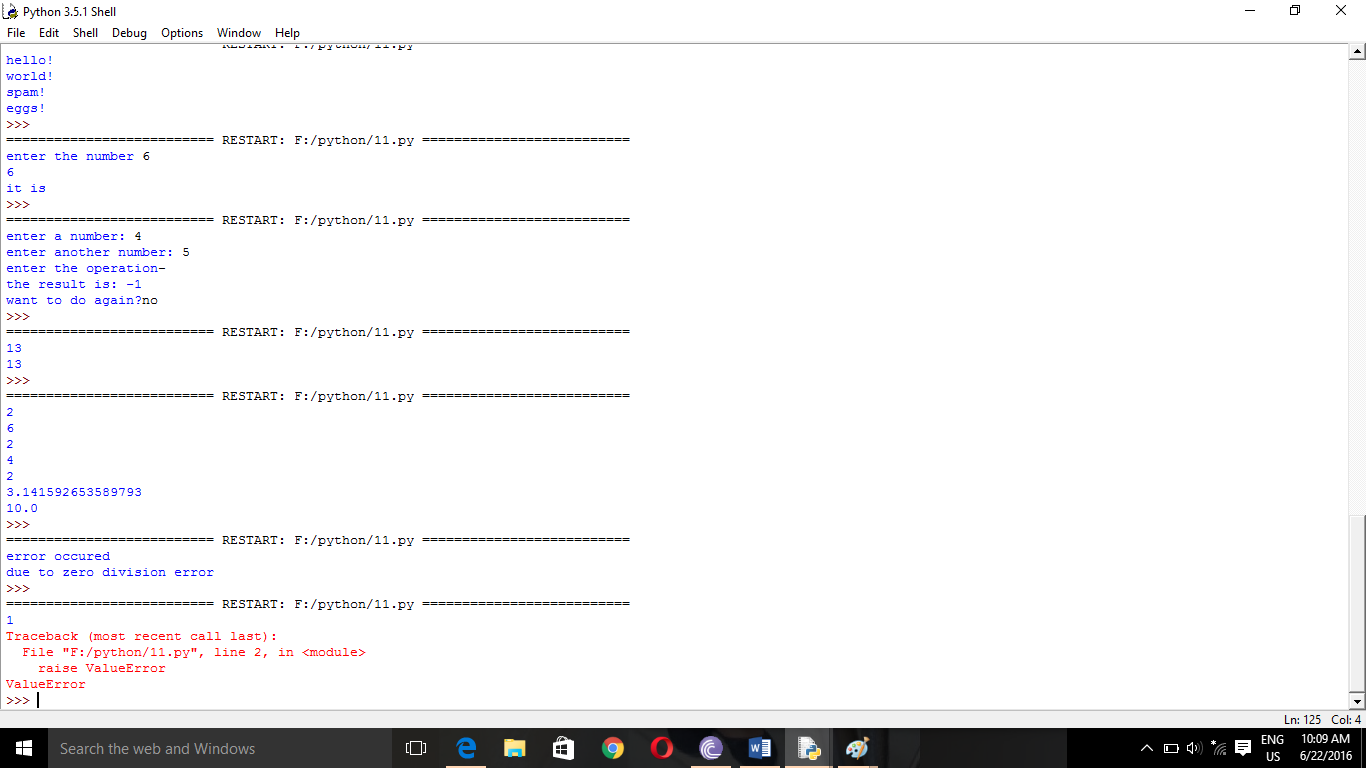
print("done calculation")

except ZeroDivisionError:

print("error occured")

print("due to zero division error")

OUTPUT:-



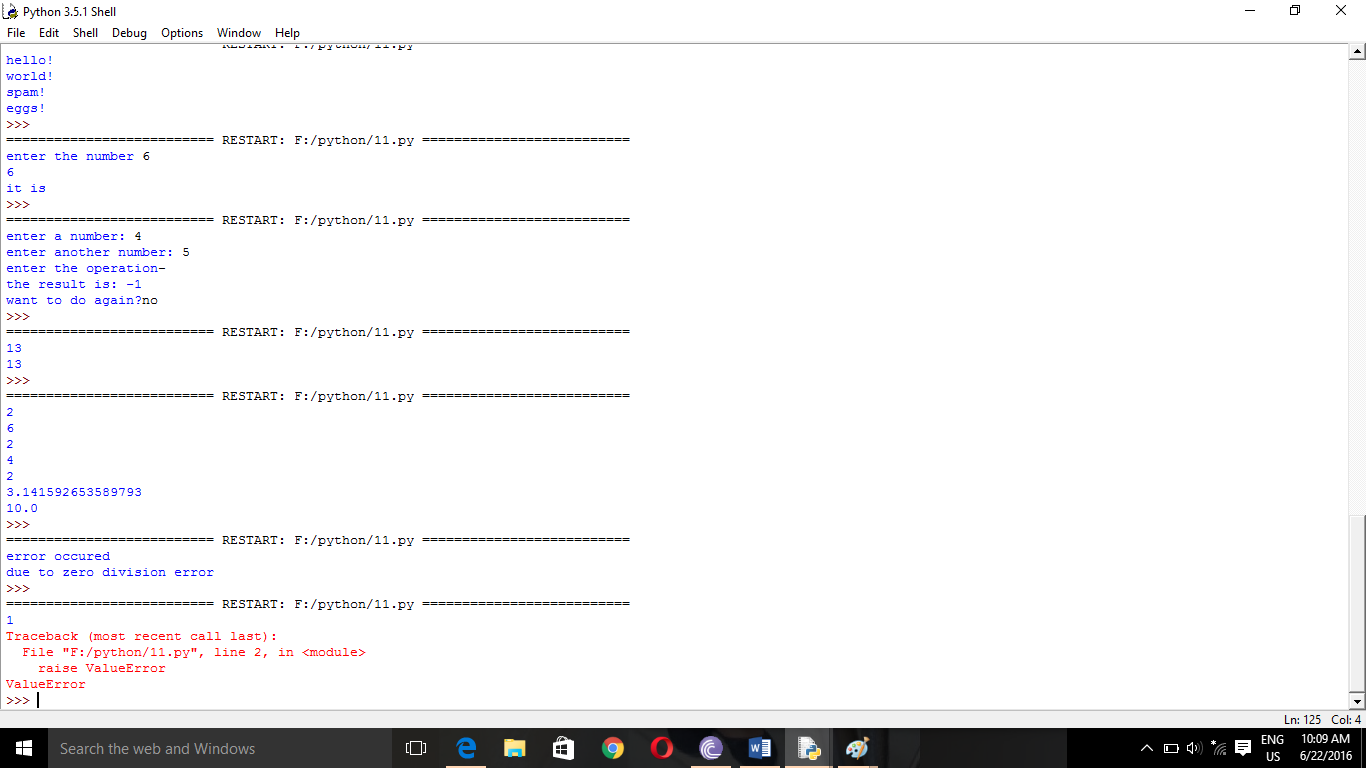
**RAISING HANDLING**

print(1)

raise ValueError

print(2)

OUTPUT:-



**19. Write a program to print table of a number.**

n=int(input('enter a no'))

i=1

while(i<11):

print(n\*i)

i=i+1

**Output:**

enter a no12

12

24

36

48

60

72

84

96

108

120

**20.Write a programe to draw Floyd’s triangle?**

c=1

for i in range (10):

res=""

for j in range(i):

res=res+str(c)

c+=1

print(res)

OUTPUT

1

23

456

78910

1112131415

161718192021

22232425262728

2930313233343536

373839404142434445

**21. Write a program to check whether a number is a Armstrong number or not .**

n=int(input("enter a number"))

i=n

sum=0

while(i!=0):

r=i%10

i=i//10

sum+=r\*\*3

if n==sum:

print("it is a amstrong")

else:

print("not ")

OUTPUT:

enter a number371

it is a amstrong

enter a number371

it is a amstrong

**22. Make a rolling dice game.**

import random

for i in range(1):

value=random.randint(1,6)

print(value)

OUTPUT: 4

**23.Write a program to change a temperature in Celsius to Fahrenheit and Fahrenheit to Celsius**

a=input("enter the type u want to convert")

if(a=='c'):

z=float(input("enter temperature in celsius "))

f=z\*(9/5)+32

print("temp in farhenheit is:",f)

elif (a=='f'):

e=float(input("enter temp in fahrenheit"))

c=e\*(5/9)-32

print("temp in celsius is",c)

OUTPUT:

enter the type u want to convertc

enter temperature in celsius 12

temp in farhenheit is: 53.6

enter the type u want to convertf

enter temp in fahrenheit15

temp in celsius is -23.666666666666664

**24.Write a progam to print a diamond pattern using asterisks.**

def run():

j = 7

k = 7

p = 1

for i in range(8):

print (" " \* k," \*" \* i)

k -=1

while j > 1:

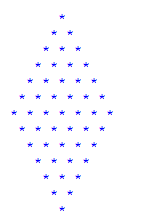
j -= 1

print (" " \* p," \*" \* j)

p +=1

run()

OUTPUT:



**25. Write a program to make a right angled triangle using asterisks.**

#right angle triangle

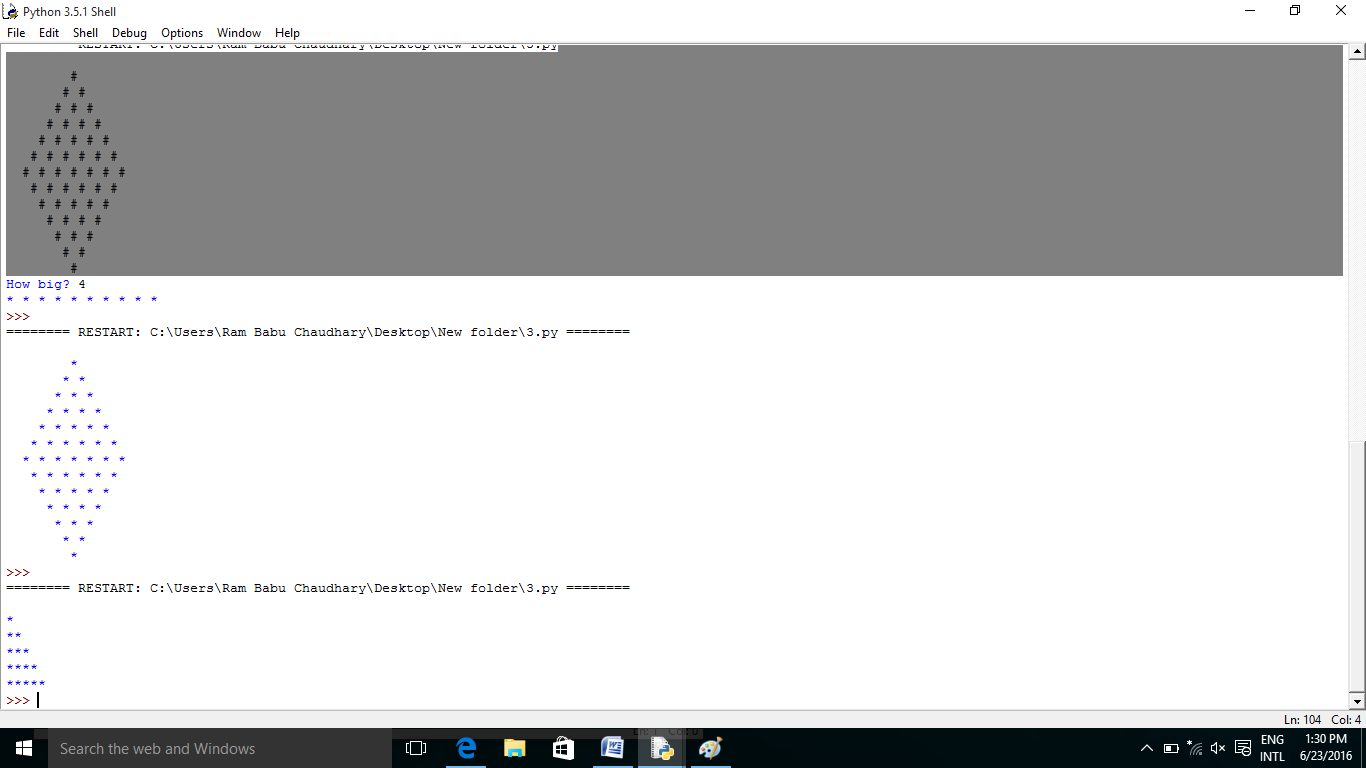
i=0

while i<6:

print(i\*"\*")

i+=1

OUTPUT:



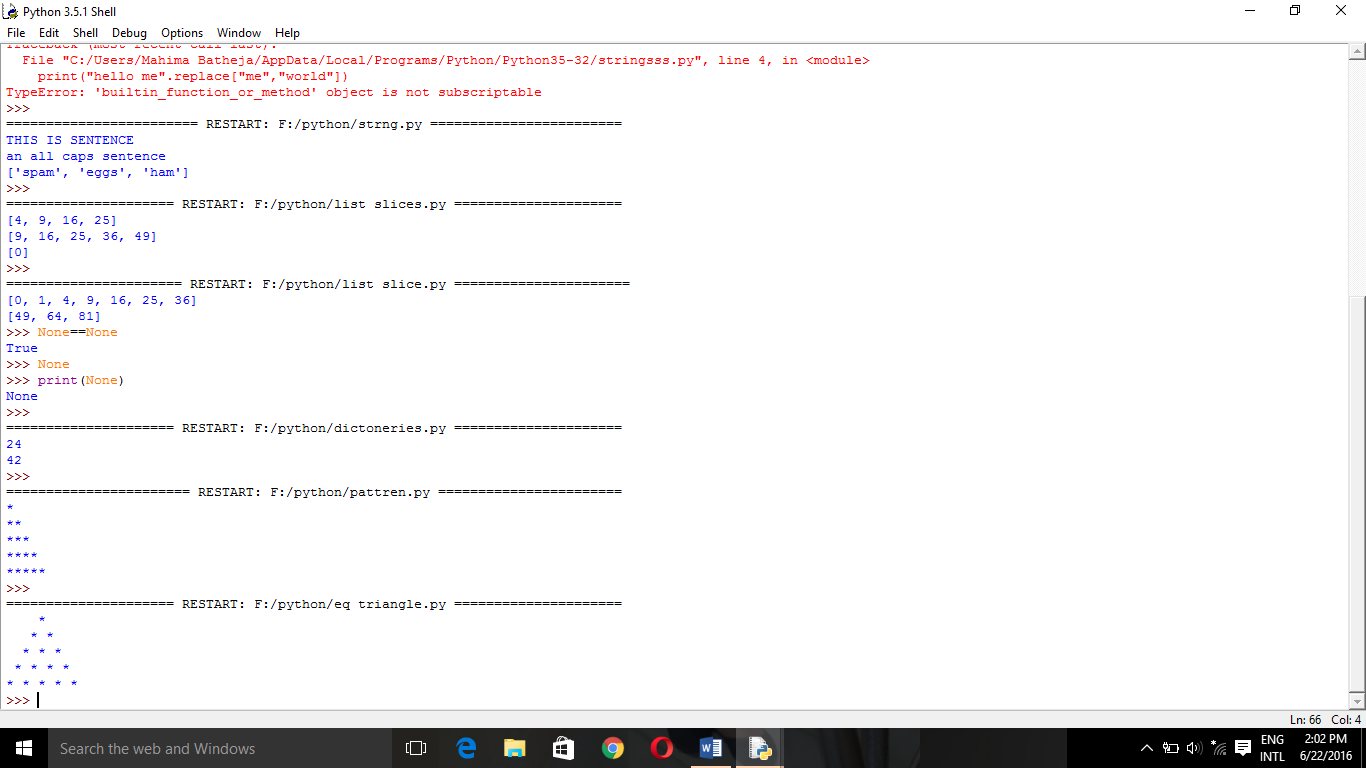
**26. Write a program to make a equilateral triangle using asterisks**

i=4

while(i!=(-1)):

print(i\*" "+(5-i)\*"\* ")

i=i-1



**27. Write a program to print Fibonacci series using recursion**.

# Python program to display the Fibonacci sequence up to n-th term using recursive functions

def recur\_fibo(n):

"""Recursive function to

print Fibonacci sequence"""

if n <= 1:

return n

else:

return(recur\_fibo(n-1) + recur\_fibo(n-2))

# take input from the user

nterms = int(input("How many terms? "))

# check if the number of terms is valid

if nterms <= 0:

print("Plese enter a positive integer")

else:

print("Fibonacci sequence:")

for i in range(nterms):

print(recur\_fibo(i))

OUTPUT:

How many terms? 9

Fibonacci sequence:

0

1

1

2

3

5

8

13

21

**28.Write a program to print Fibonacci series.**

z=int(input("enter a number"))

k=1

j=1

print(k)

print(j)

sum=0

for i in range(z-2):

sum=k+j

k=j

j=sum

print(sum)

**OUTPUT:**

enter a number12

1

1

2

3

5

8

13

21

34

55

89

144

**29.Write a program to print pascals triangle**

def scan(op, seq, it):

a = []

result = it

a.append(it)

for x in seq:

result = op(result, x)

a.append(result)

return a

def pascal(n):

def nextrow(row, x):

return [l+r for l,r in zip(row+[0,],[0,]+row)]

return scan(nextrow, range(n-1), [1,])

for row in pascal(10):

print(row)

[1]

[1, 1]

[1, 2, 1]

[1, 3, 3, 1]

[1, 4, 6, 4, 1]

[1, 5, 10, 10, 5, 1]

[1, 6, 15, 20, 15, 6, 1]

[1, 7, 21, 35, 35, 21, 7, 1]

[1, 8, 28, 56, 70, 56, 28, 8, 1]

[1, 9, 36, 84, 126, 126, 84, 36, 9, 1]

>>>

**30. CHECK WHETHER YEAR IS LEAP OR NOT**

year=int(input("ënter the year"))

if(year%4==0):

print("year is leap")

else:

print("not")

OUTPUT:

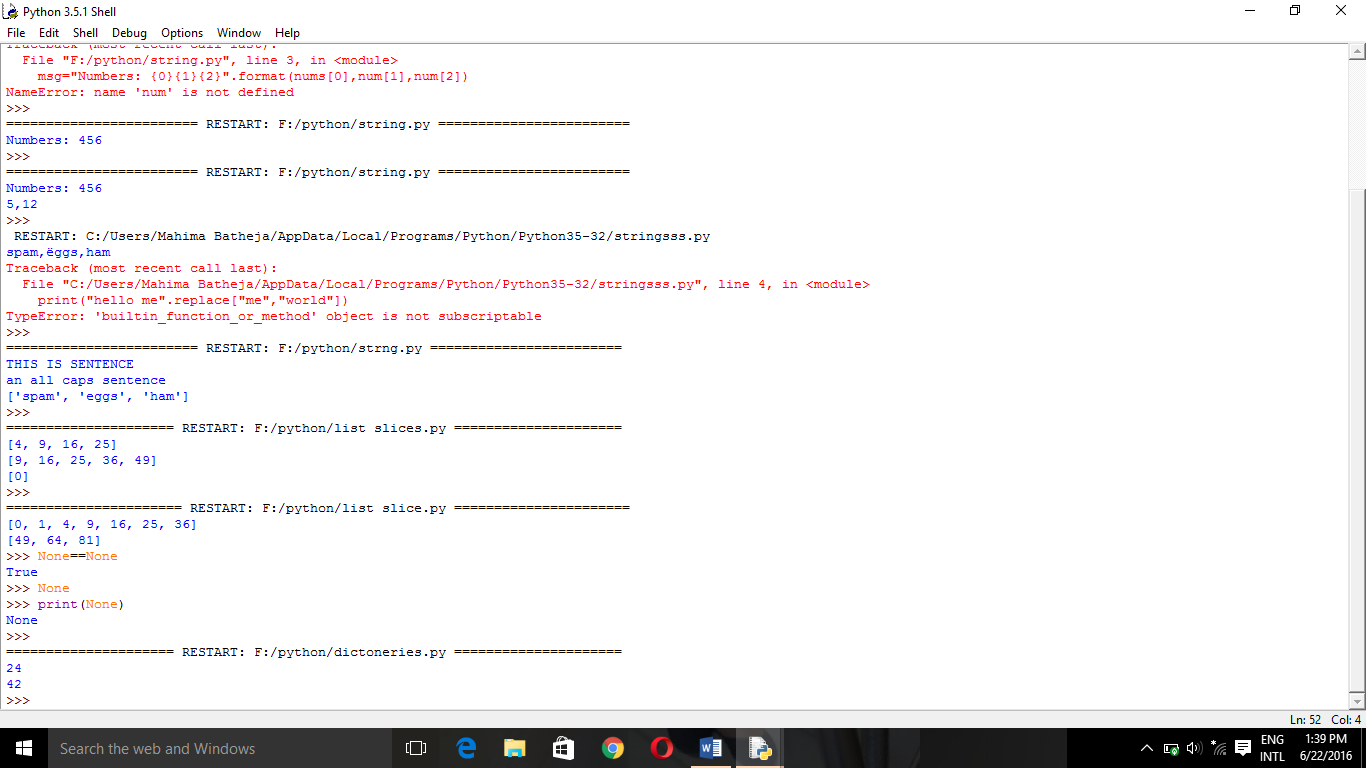
**31. DICTIONARIES**

ages = {"Dave":24,"Mary":42,"john":58}

print(ages["Dave"])

print(ages["Mary"])

OUTPUT:-



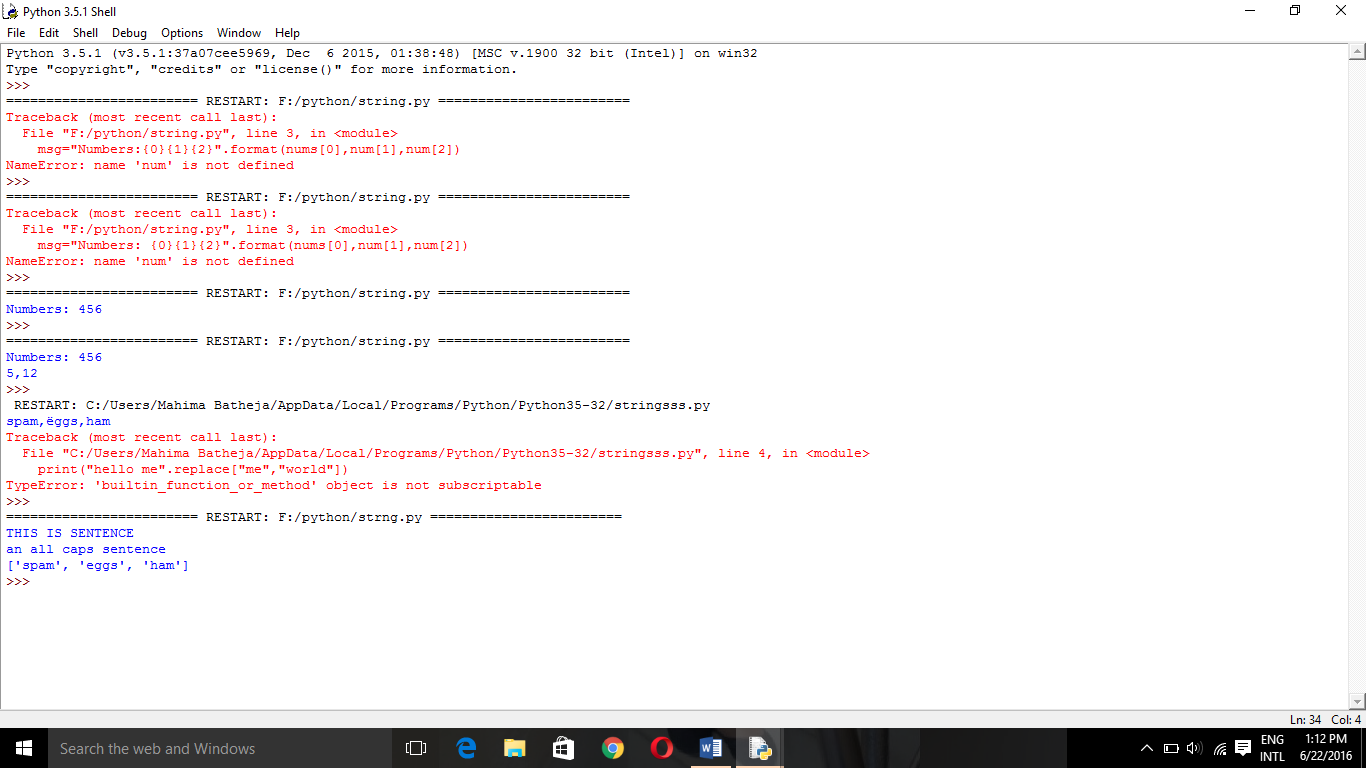
**32. STRING FUNCTION**

print("this is sentence".upper())

print("AN ALL CAPS SENTENCE".lower())

print("spam,eggs,ham".split(","))

OUTPUT:-



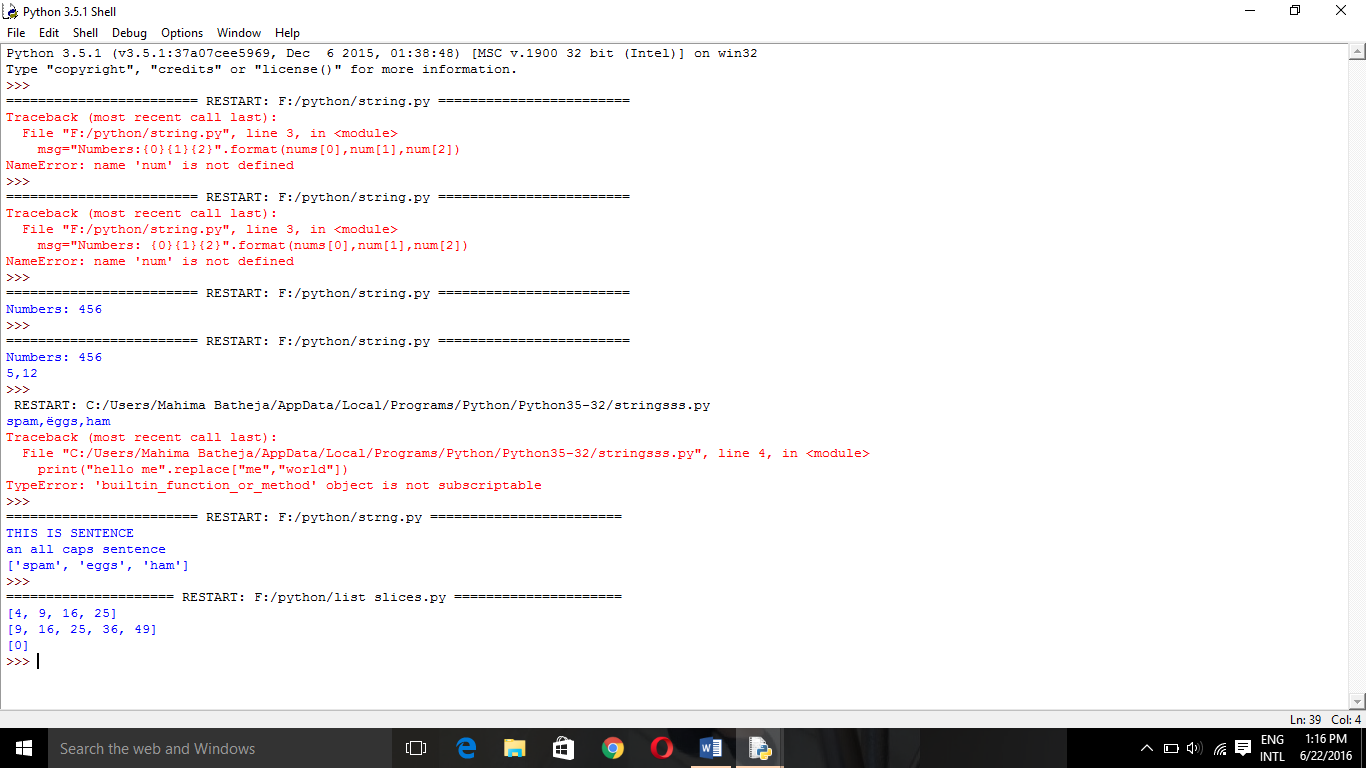
**33. LIST SLICES**

squares=[0,1,4,9,16,25,36,49,64,81]

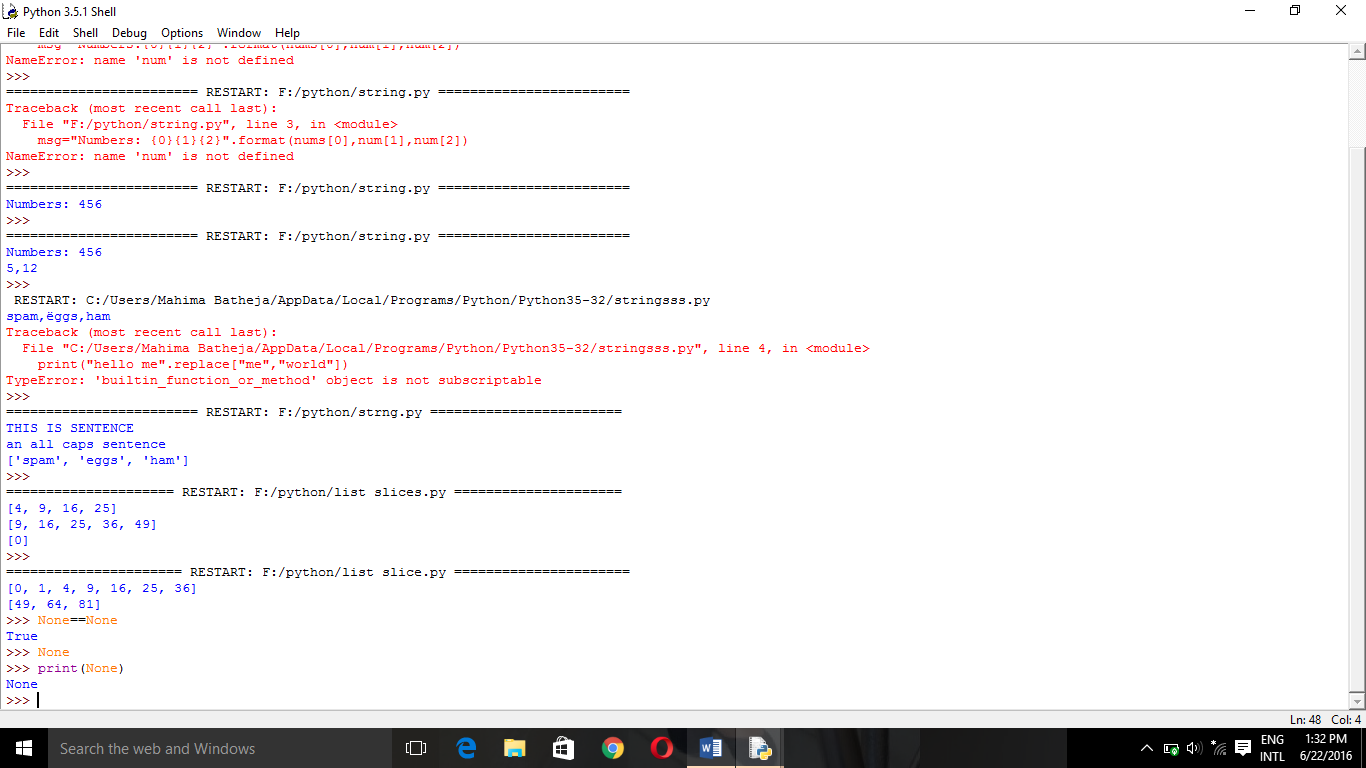
print(squares[2:6])

print(squares[3:8])

print(squares[0:1])

OUTPUT:- 

**34.NONE**



**35.CHECK WHETHER THE NUMBER IS PALINDROME OR NOT**

n=int(input("enter the number"))

i=n

sum=0

while(i!=0):

r=i%10

i=i//10

sum=sum\*10+r

print(sum)

if(sum==n):

print("it is a palindrome")

else:

print("not a palindrome")

OUTPUT:

enter the number696

696

it is a palindrome

**CALCULATOR**

ch='y'

while(ch=='y' or ch=='Y'):

n=int(input("enter a number: "))

m=int(input("enter another number: "))

op=input("enter the operation")

if(op=='+'):

res=n+m

print("the result is: "+str(res))

elif(op=='-'):

res=n-m

print("the result is: "+str(res))

elif(op=='%'):

res=n%m

print("the result is: "+str(res))

elif(op=='\*'):

res=n\*m

print("the result is: "+str(res))

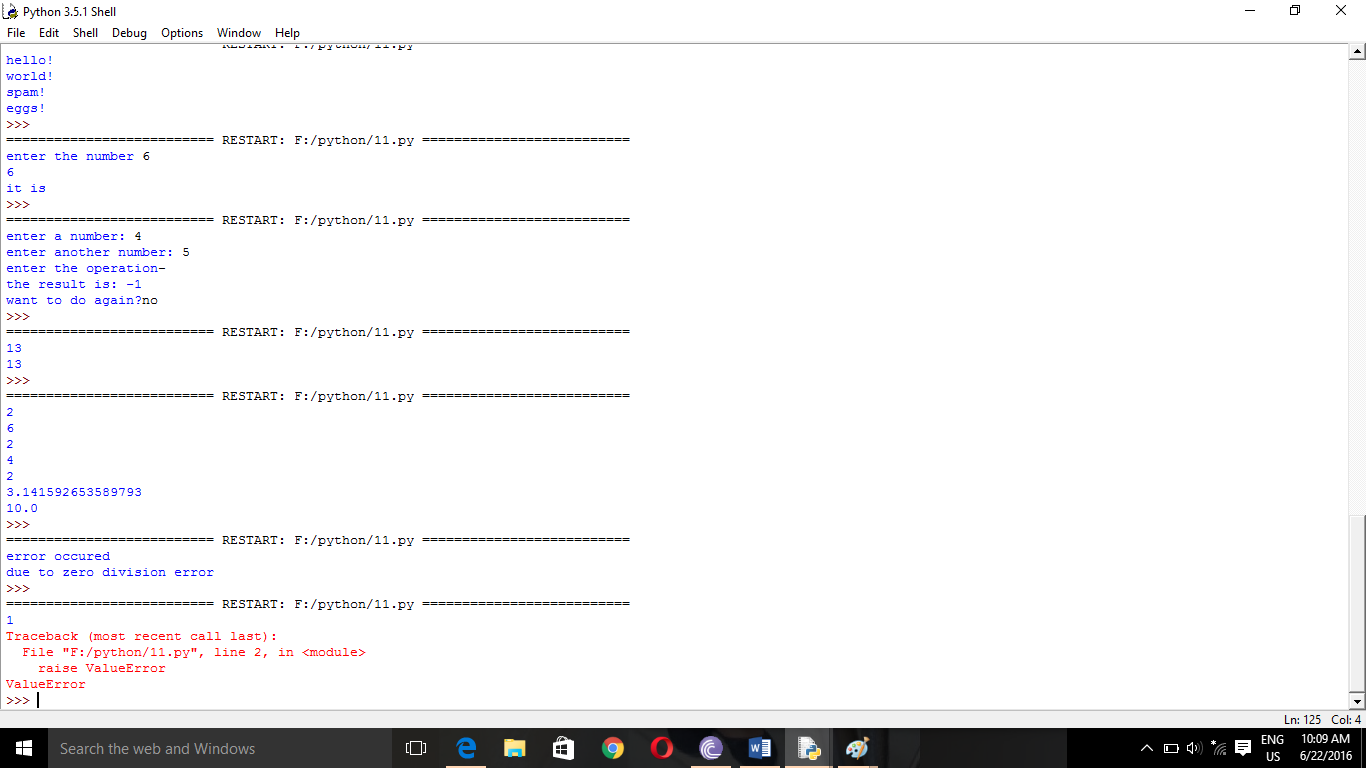
else:

print("wrong input")

break

ch=input("want to do again?")

OUTPUT:-



**MULTIPLICATION TABLE**

print("MULTIPLICATION TABLE")

for i in range(1,11):

print(i,end="\t")

print()

for j in range(1,11):

for k in range(1,11):

print(j\*k,end="\t")

print("\n")

MULTIPLICATION TABLE

1 2 3 4 5 6 7 8 9 10

1 2 3 4 5 6 7 8 9 10

2 4 6 8 10 12 14 16 18 20

3 6 9 12 15 18 21 24 27 30

4 8 12 16 20 24 28 32 36 40

5 10 15 20 25 30 35 40 45 50

6 12 18 24 30 36 42 48 54 60

7 14 21 28 35 42 49 56 63 70

8 16 24 32 40 48 56 64 72 80

9 18 27 36 45 54 63 72 81 90

10 20 30 40 50 60 70 80 90 100