**FEATURES**

**Easy to Learn**

**OOP : Reusability**

**FREE and Open Source**

**Career Perspective**

**dJango (Website Developer)**

**AWS, Devops, (Automation Tools)**

**Data Science, Tableau, Big data (Data Analyst)**

**Networking, EH, Security**

**Testing ( Selenium, QTP)**

**Python : 35 to 40 Hours ( 2 Months)**

**Daily : 9.30 to 11.30 : One Month : Mon to fri**

**Python Basics: 4 Hours : 11.30 to 12.30 pm**

**Data Types, Variable, IF, Loops, Functions, collections**

**Python Training::**

**String, Files, Regular Expressions**

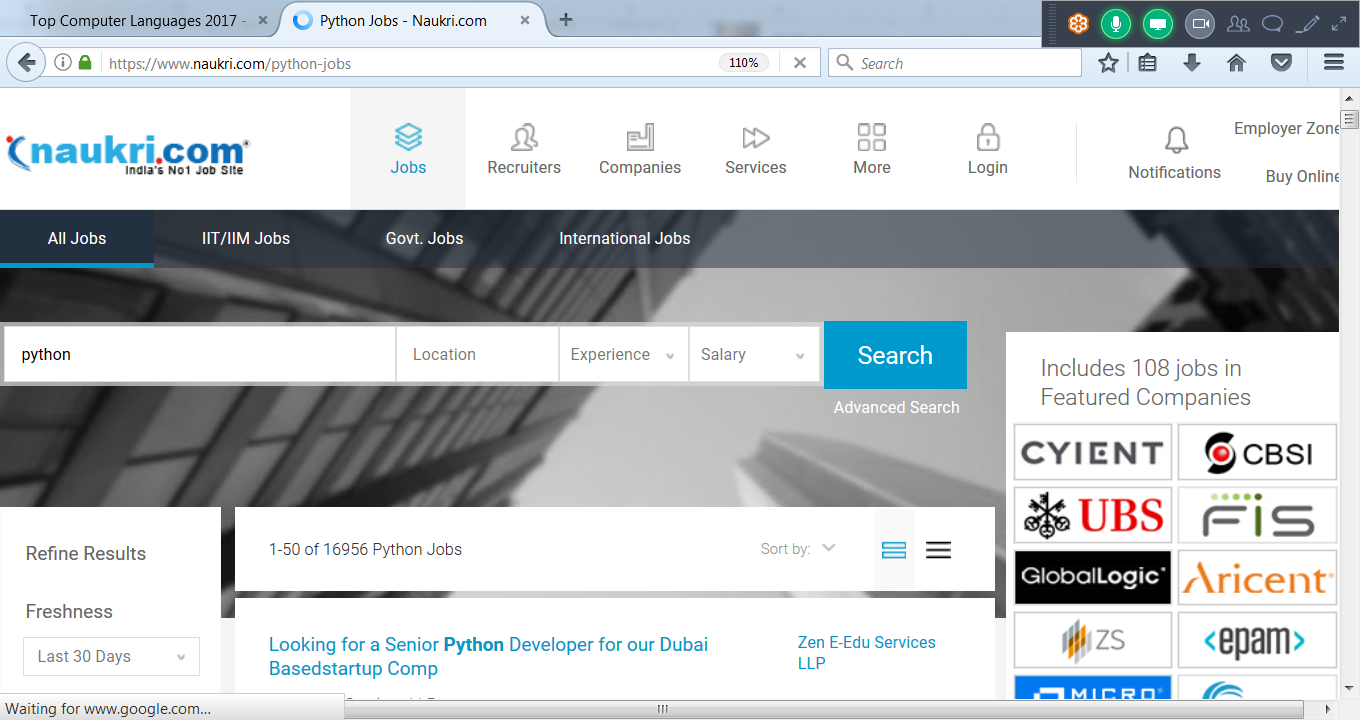
**Exceptions, Debugging, Function,**

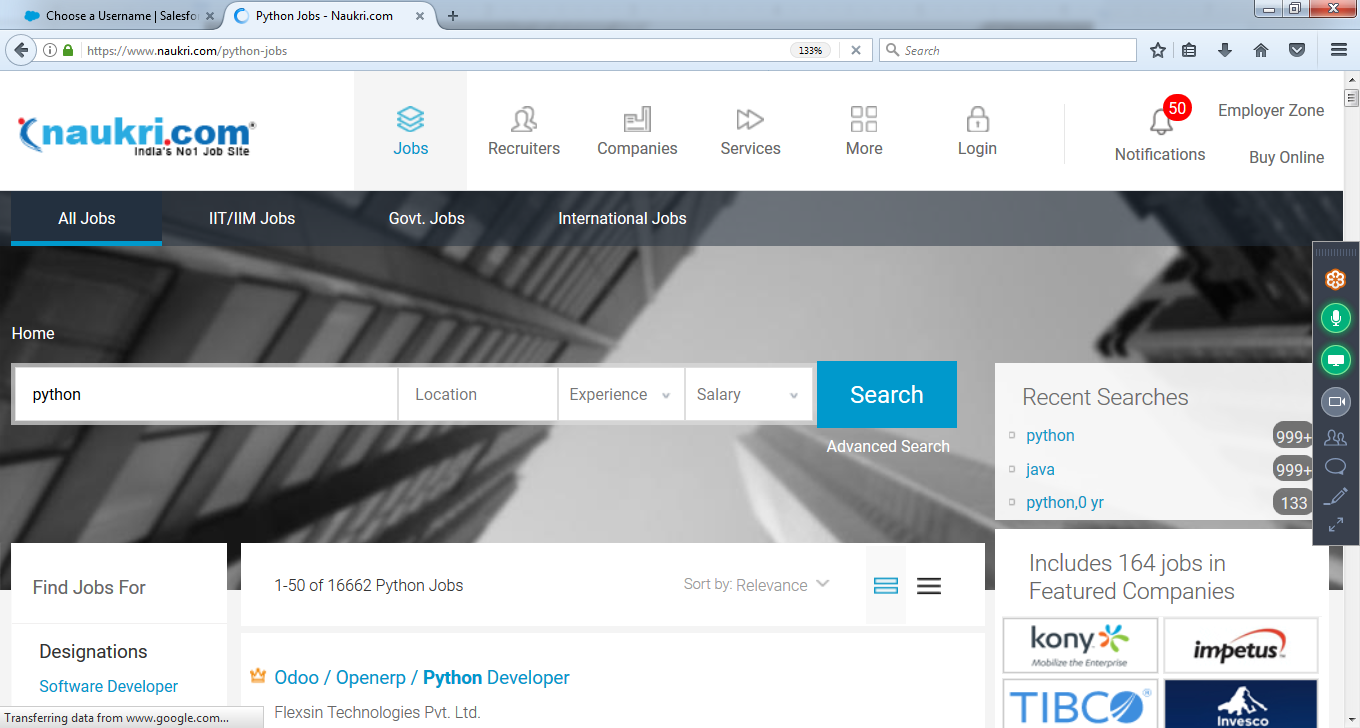
**List, Dictionary, OOPS, Generators and Decorators**

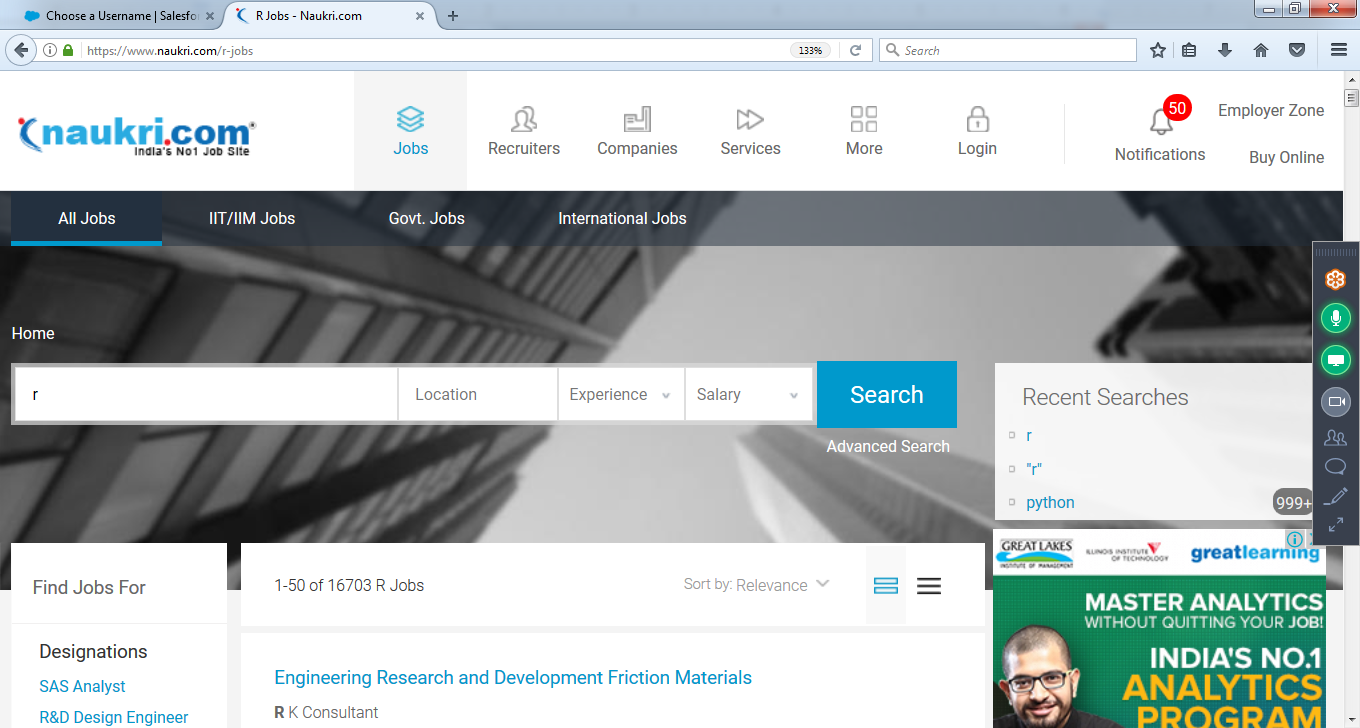
**Multithreading, GUI applications, Database**

**http://statisticstimes.com/tech/top-computer-languages.php**

****

****

****

****

**Scripting** Uses **PYTHON ( Java Script, Angular JS)**

**Web Technology django** uses **PYTHON (Java, .net, php)**

**Flask (Mobile Based)**

**Reporting Tableau**

**Data Science** Analytics uses **PYTHON (R)**

**Big Data** **Hadoop** uses **PYTHON ( Java)**

Amazon **Web Services** uses **PYTHON (Python)**

**Automation Tools** **Devops** uses **PYTHON**

**Gaming, Scientific Applications : Numpy, Scipy**

**Advantages : Easy to Learn,**

**Open Source, Web Tech**

**Google, dropbox, NASA,**

**Facebook, youtube**

* **Google - Python is one of the key language used in google.**
* **Philips - Philips uses Python for the sequencing language ( language that tells what steps each robot should take ).**
* **Frequentis - Frequentis is the originator of TAPTools, a software product that is used for air traffic control in many airports. This particular tool provides updates on the weather and runway conditions to air traffic controllers. So, you depend on Python when you fly.**
* **Quora - Quora also chose Python for its development**
* **CodesDope - Yes, we also use Python.**
* **Battlefield 2 - Battlefield 2 uses Python to implement core elements of its gameplay such as score-keeping and team-balancing.**
* **Instagram - Instagram also uses Python for its backend**
* **Walt Disney Feature Animation - Walt Disney Feature Animation is also using Python to make their animation production system more efficient in scripting.**
* **NASA - Johnson Space center uses Python in its Integrated Planning System as the standard scripting language.**
* **Civilization 4 - The new addition to the legendary strategy game series has all of its inner logic, including AI, implemented in Python.**
* **YouTube - We all love youtube and and youtube loves using Python**
* **DropBox - Dropbox allows us to store, sync, and share almost anything using the power of Python.**
* **Pinterest - Pinterest is a visual discovery tool that allows users to showcase their interests through the posting of pictures.**

**Programing Basics at 11 am**

**1 Installation, Expression, Data types, Variables, Execution**

**2 Operators and Flow Controls ( If, If-else)**

**3 Loops : for, while, Range( Examples)**

**4 Functions Collections( List, tuple, Dictionary)**

**Course Training::**

**Python : Strings**

**Python : Regular Expressions**

**Python : Files**

**Python: Debugging, Exceptions**

**Python : Functions**

**Python : Collections (List, set , Dictionary)**

**python : OOPS**

**python : Generators and Decorators**

**python : Multithreading, GUI, Games**

**Data Science**

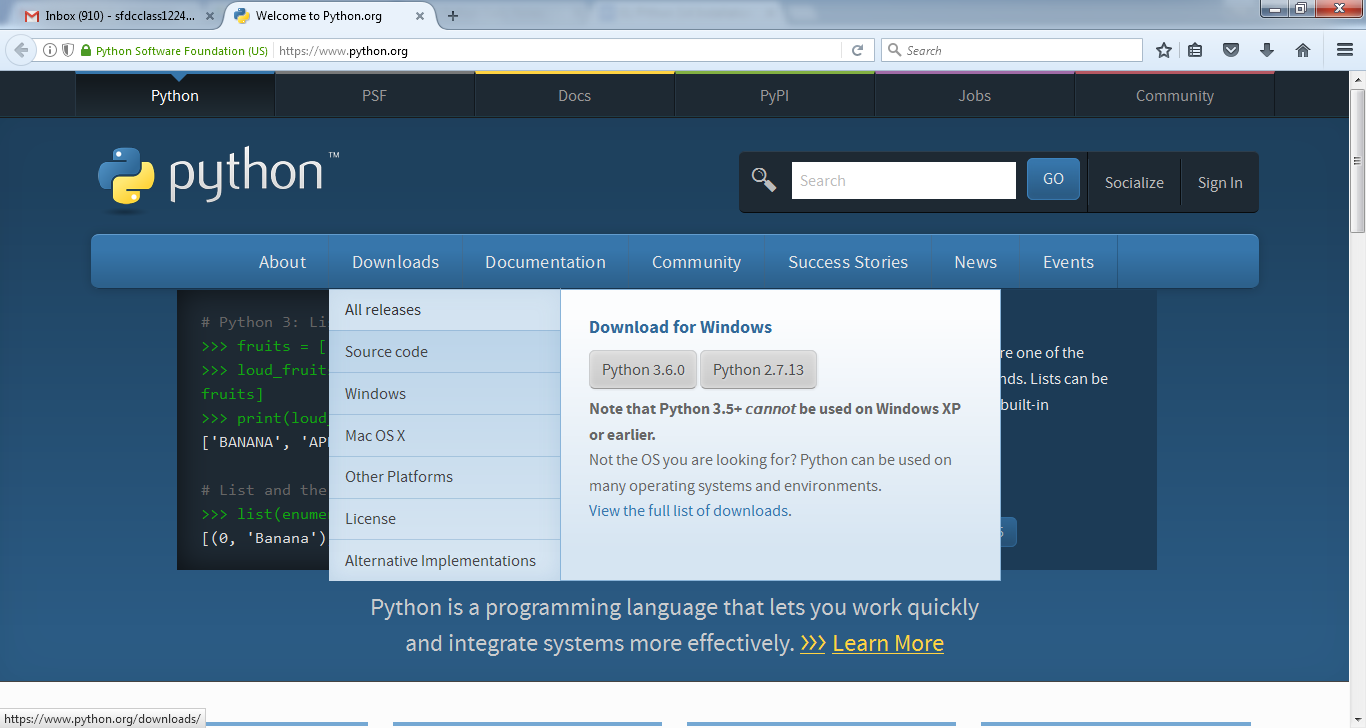
**Python, R, Machine Learning**

**Deep Learning, AI, PRed anlay**

**Python Installation**

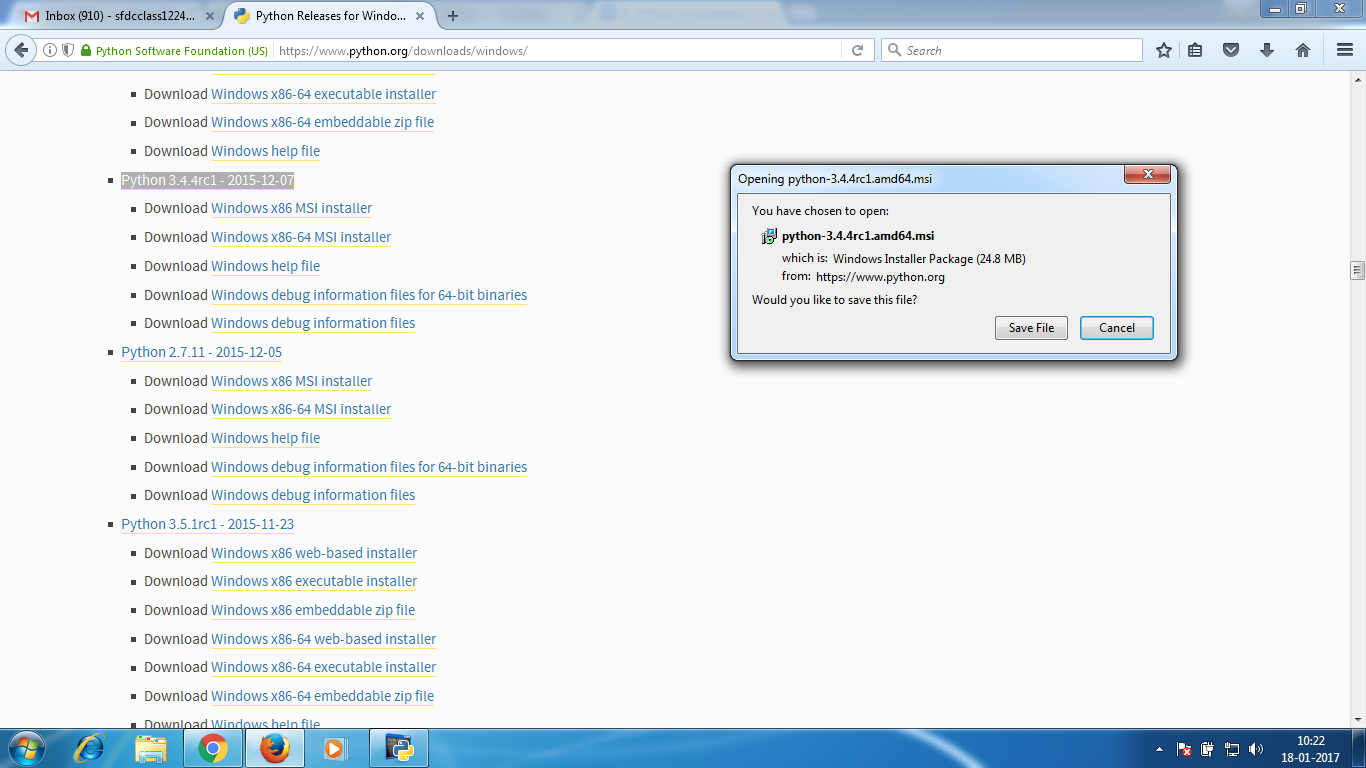
**Goto python.org**

**https://www.python.org**

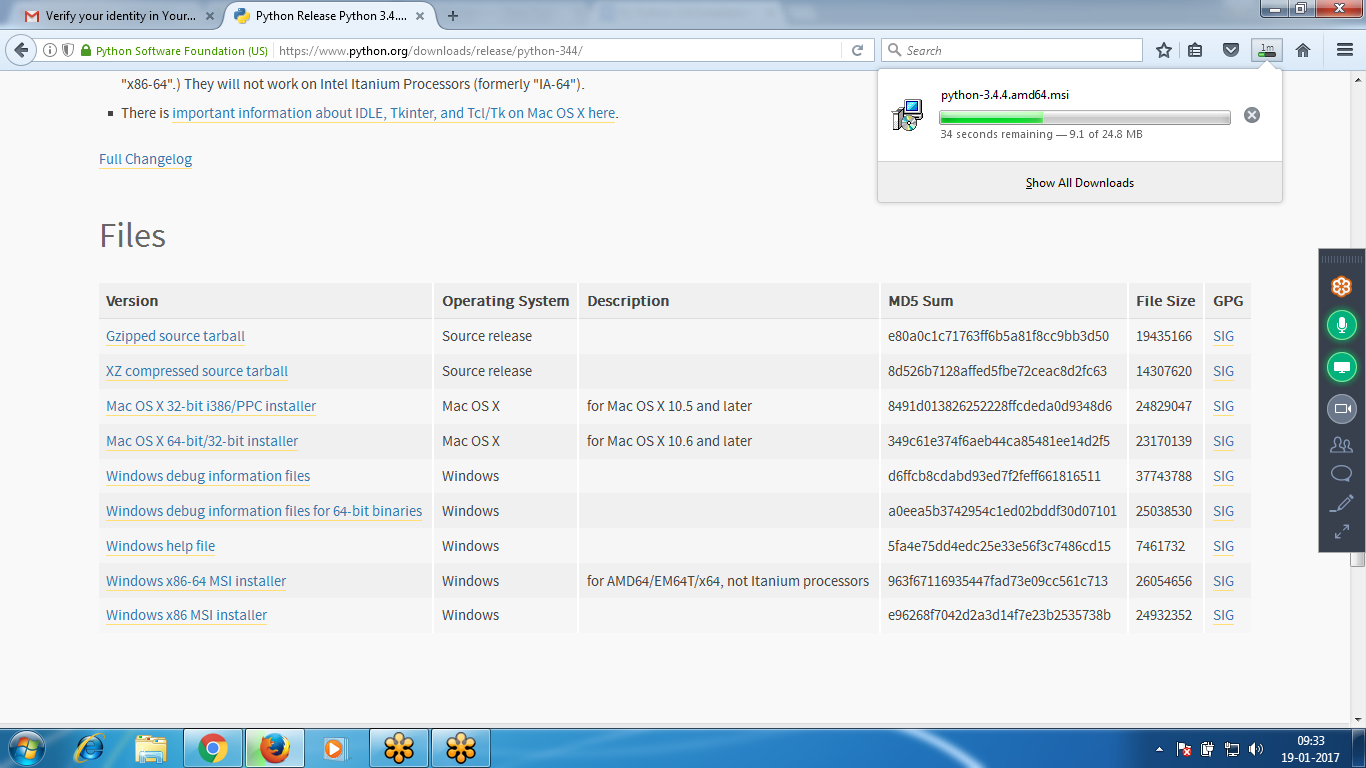
****

[**https://www.python.org/downloads/release/python-344rc1/**](https://www.python.org/downloads/release/python-344rc1/)

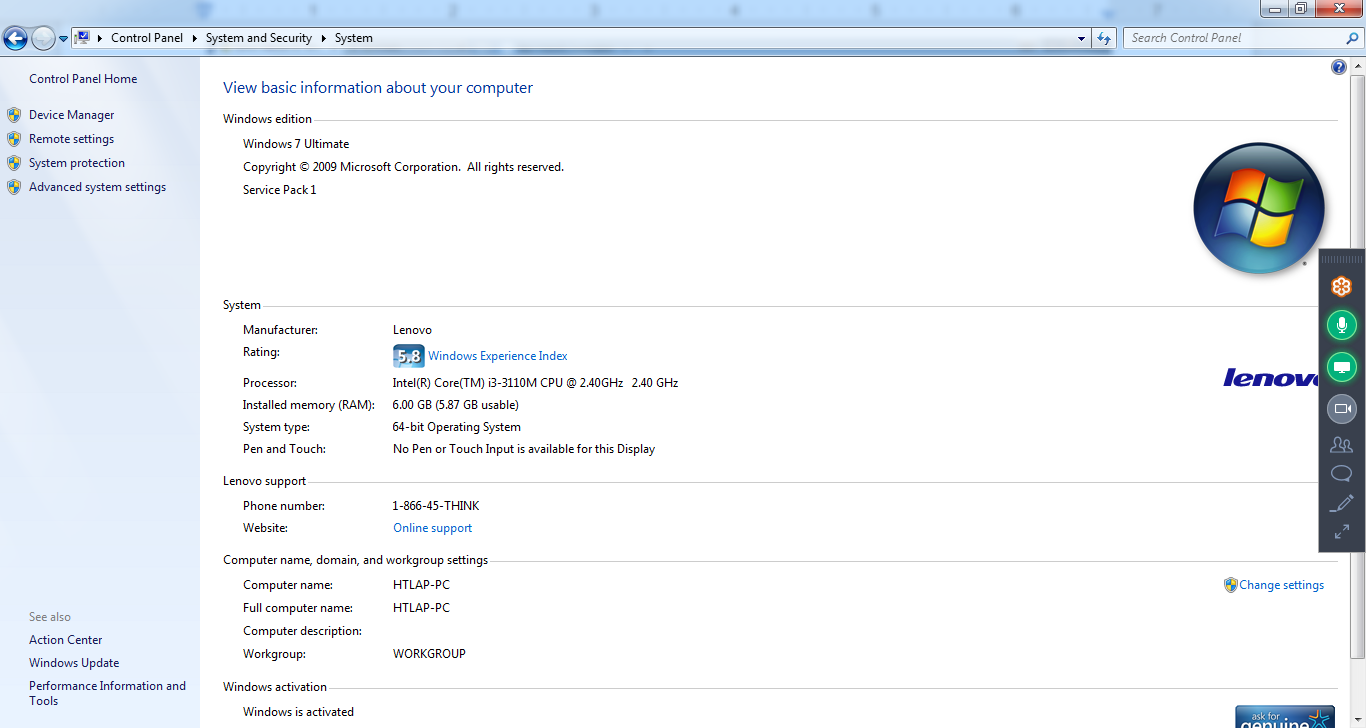
**Select** [**Python 3.4.4rc1 - 2015-12-07**](https://www.python.org/downloads/release/python-344rc1/) **- MSI Installer**



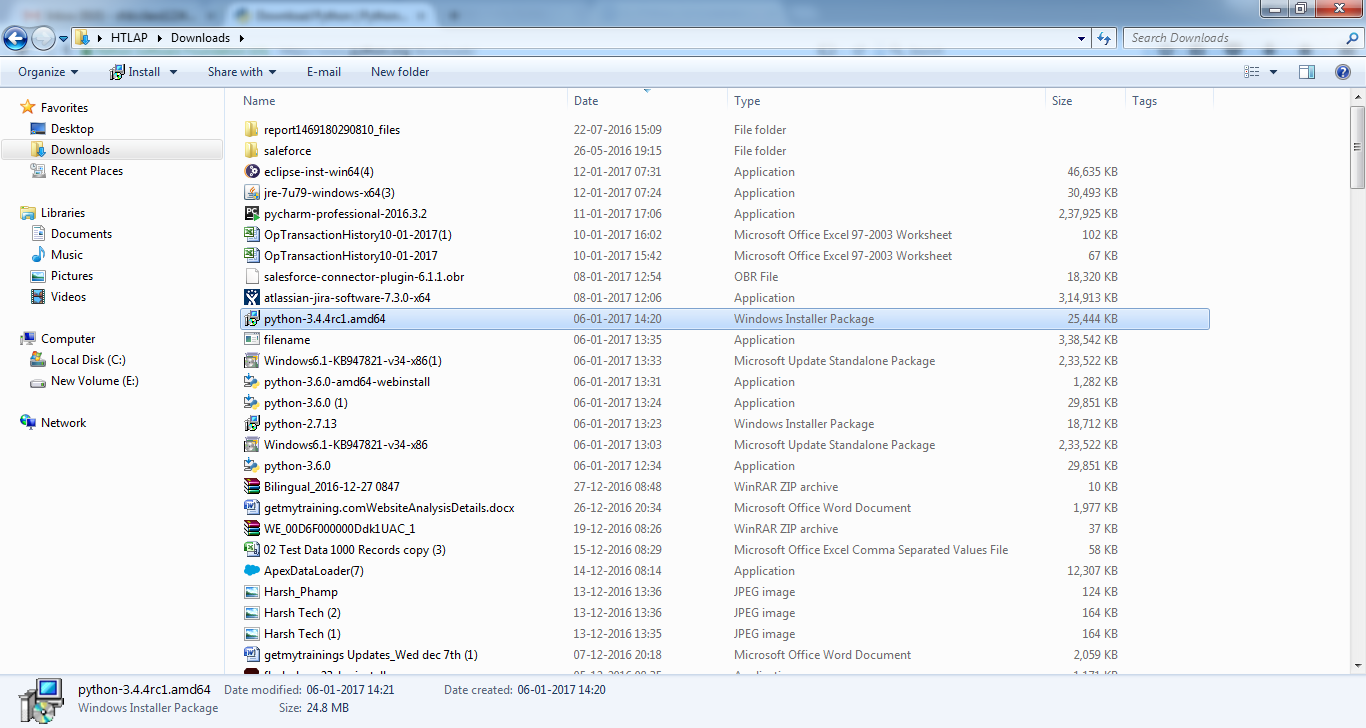
Select MSI Installer (64 Bit) or 32 Bit



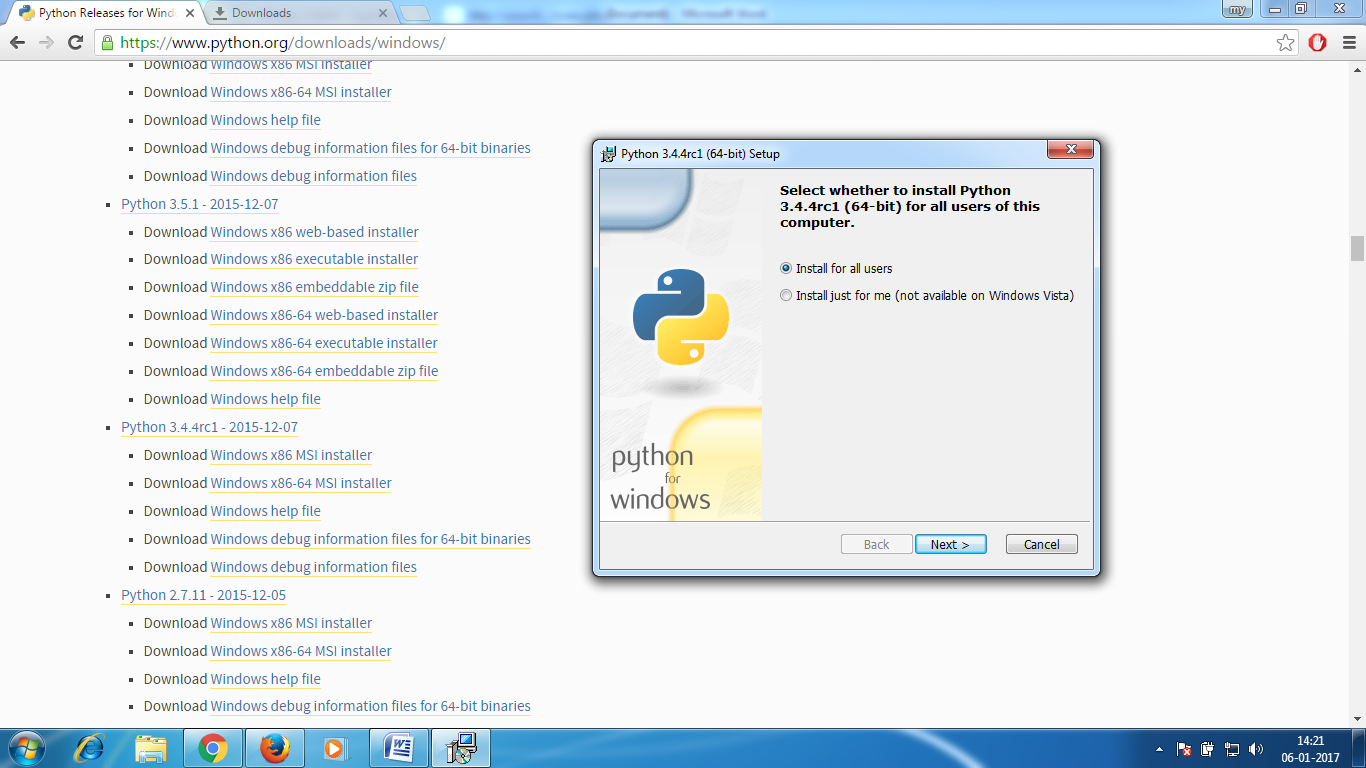
**How to IDentify 64 Bit : Control panel - system and security - System**



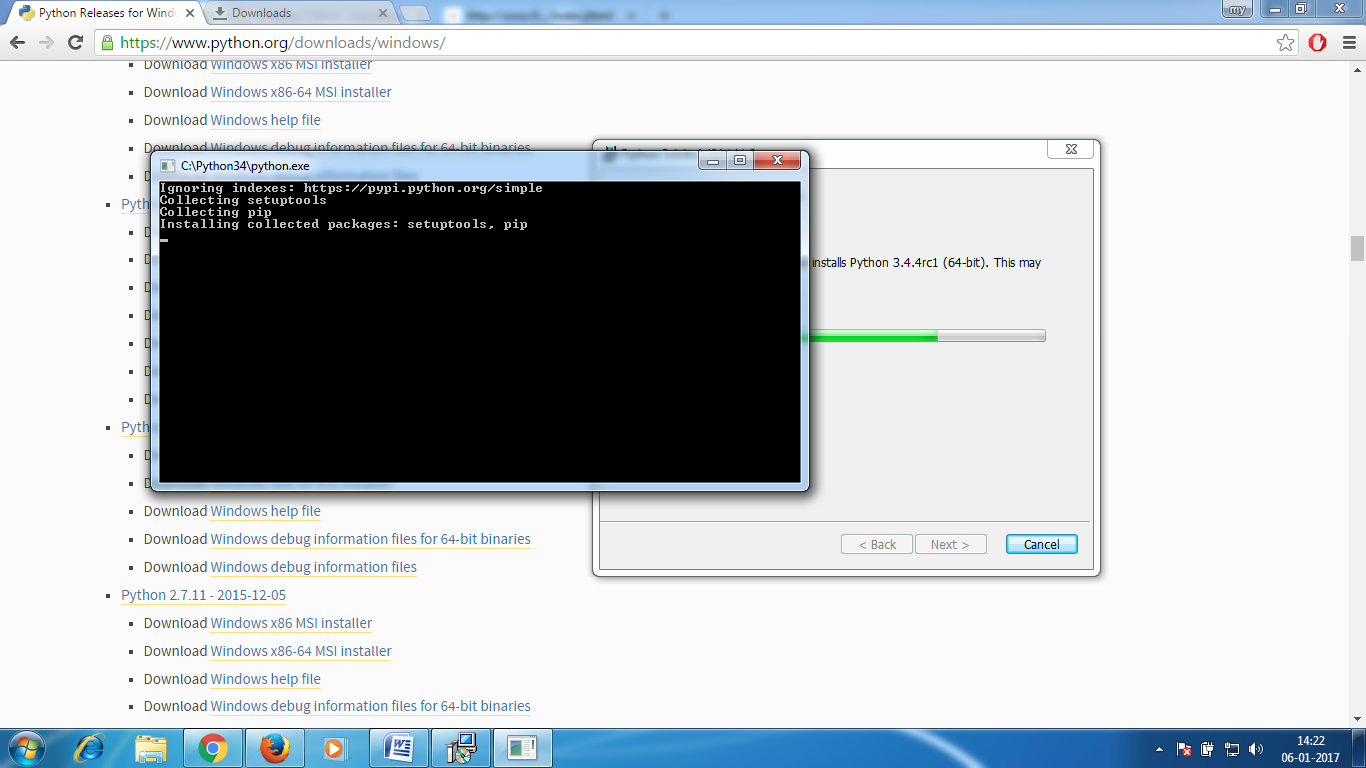
Goto Downloads :: select python 3.4.4rc1

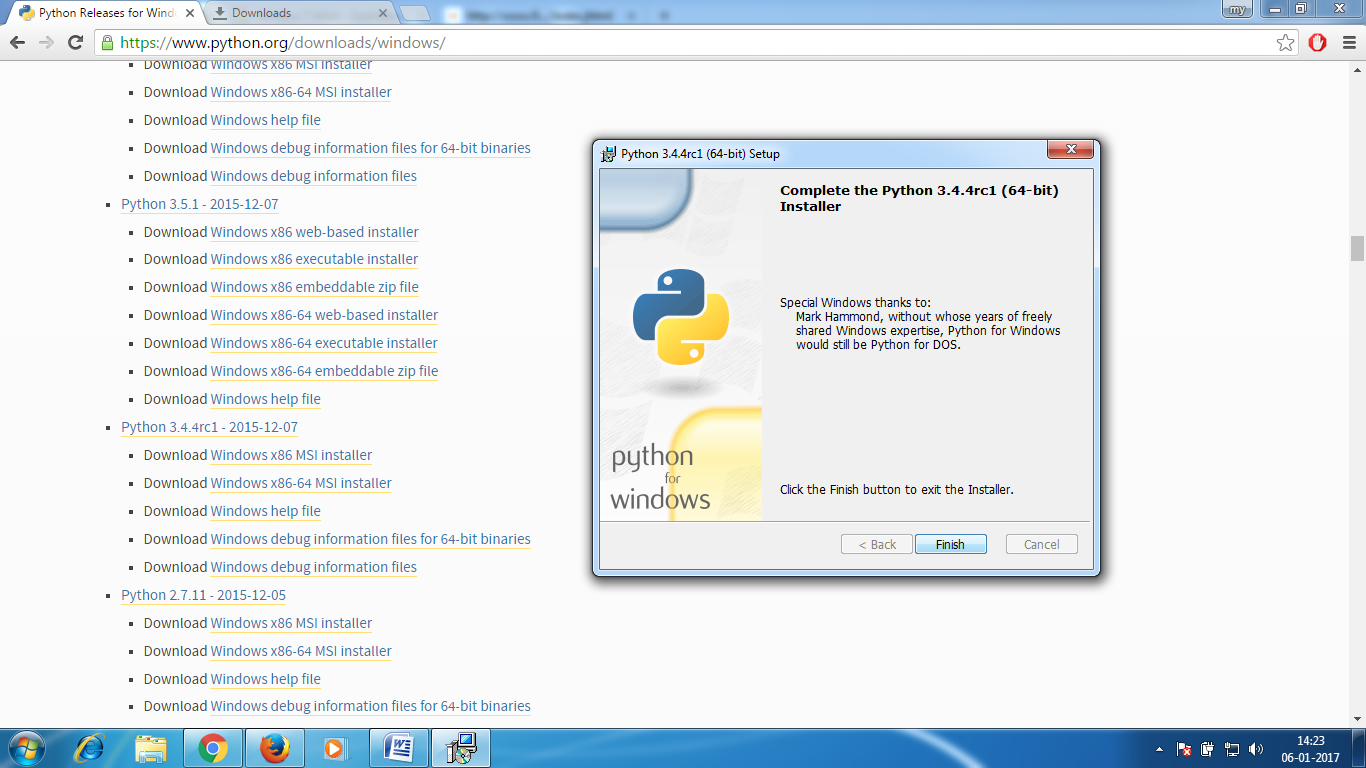
****

**Install for all users - Next**



Press Next

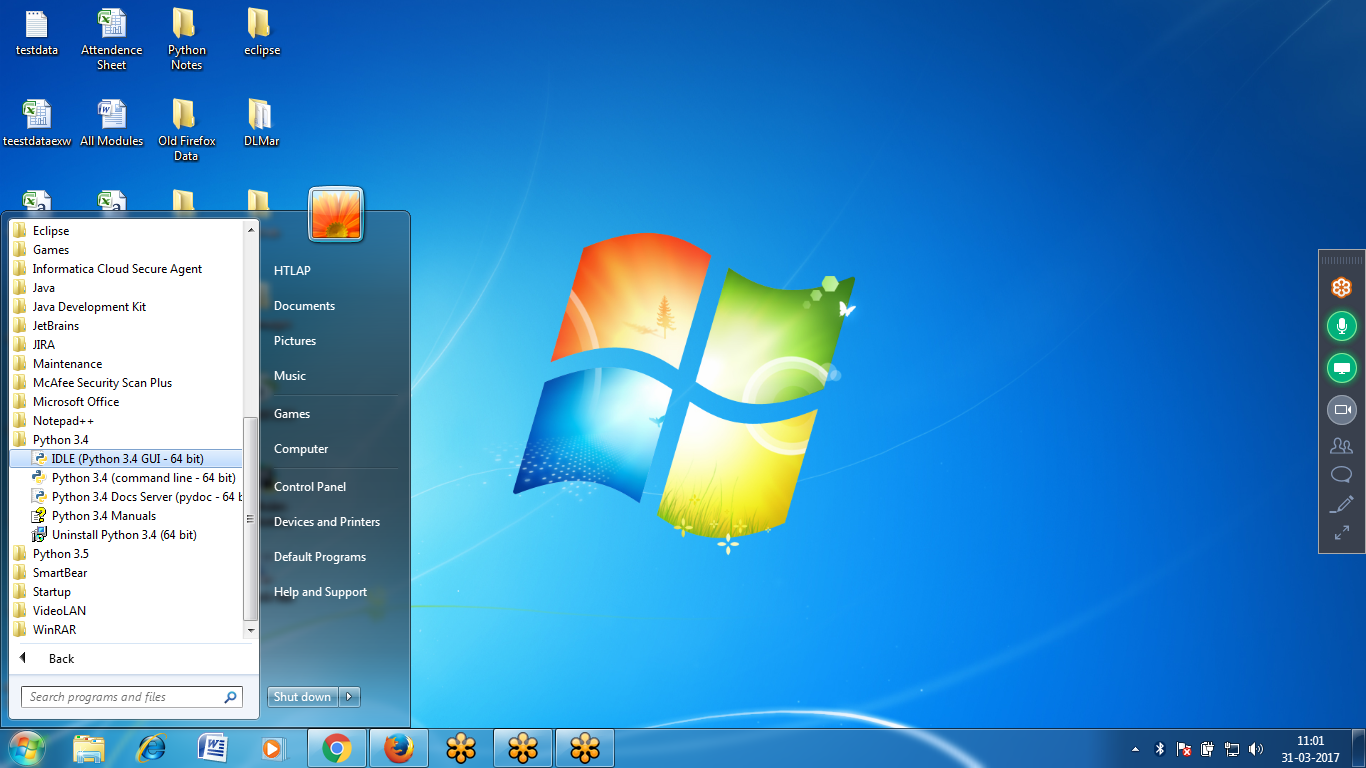




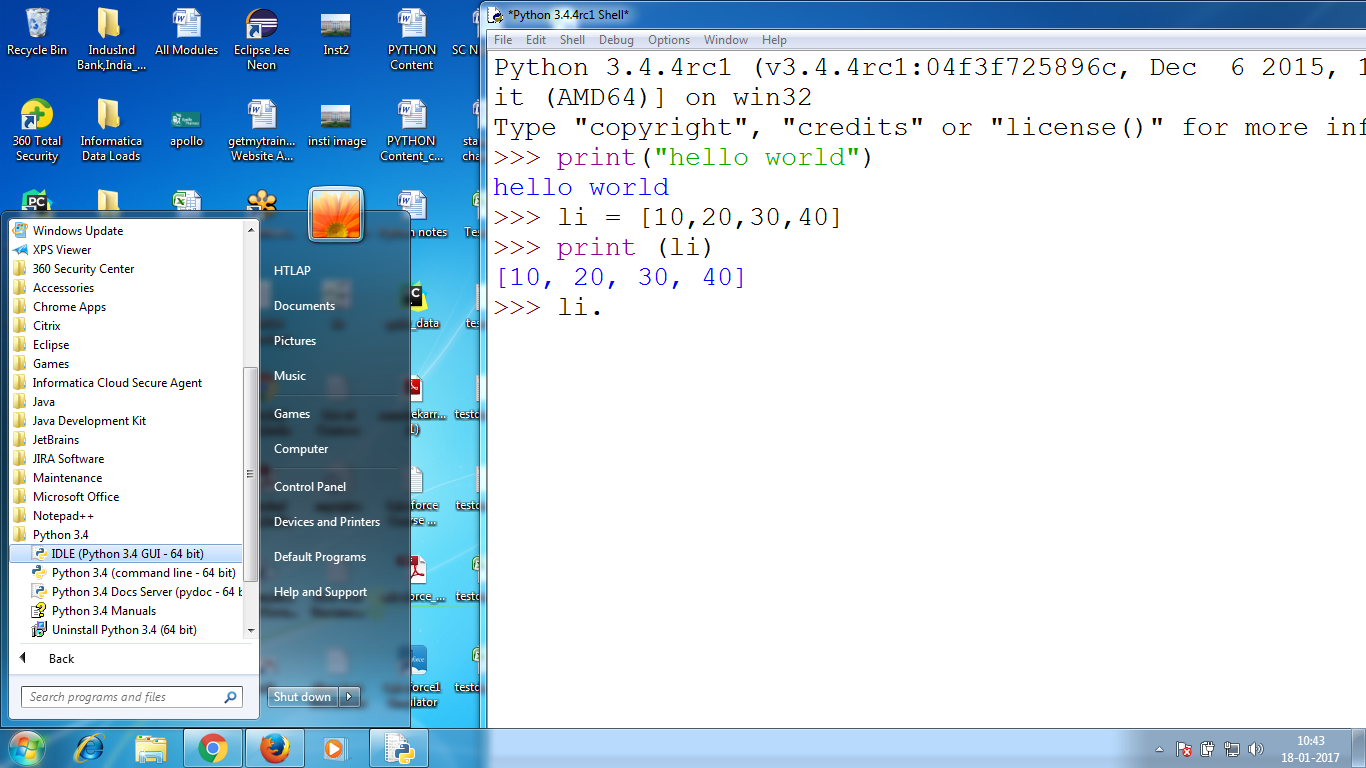
**IDLE :** python Editor

Pycharm, Anaconda, Jupyter : PYTHON

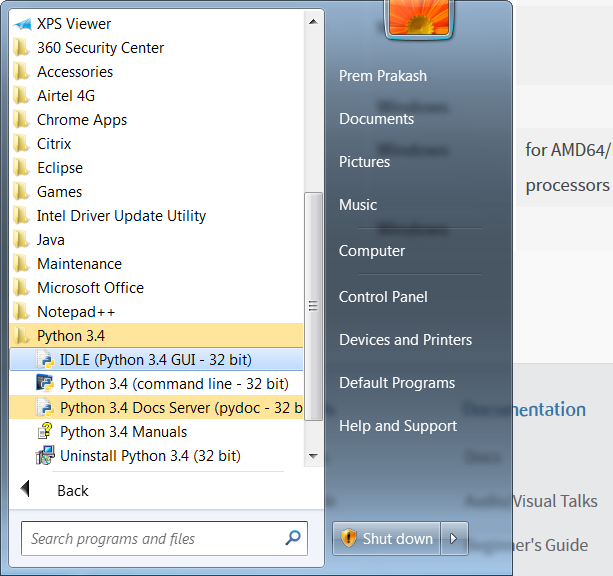
Eclipse : Java Based Editor



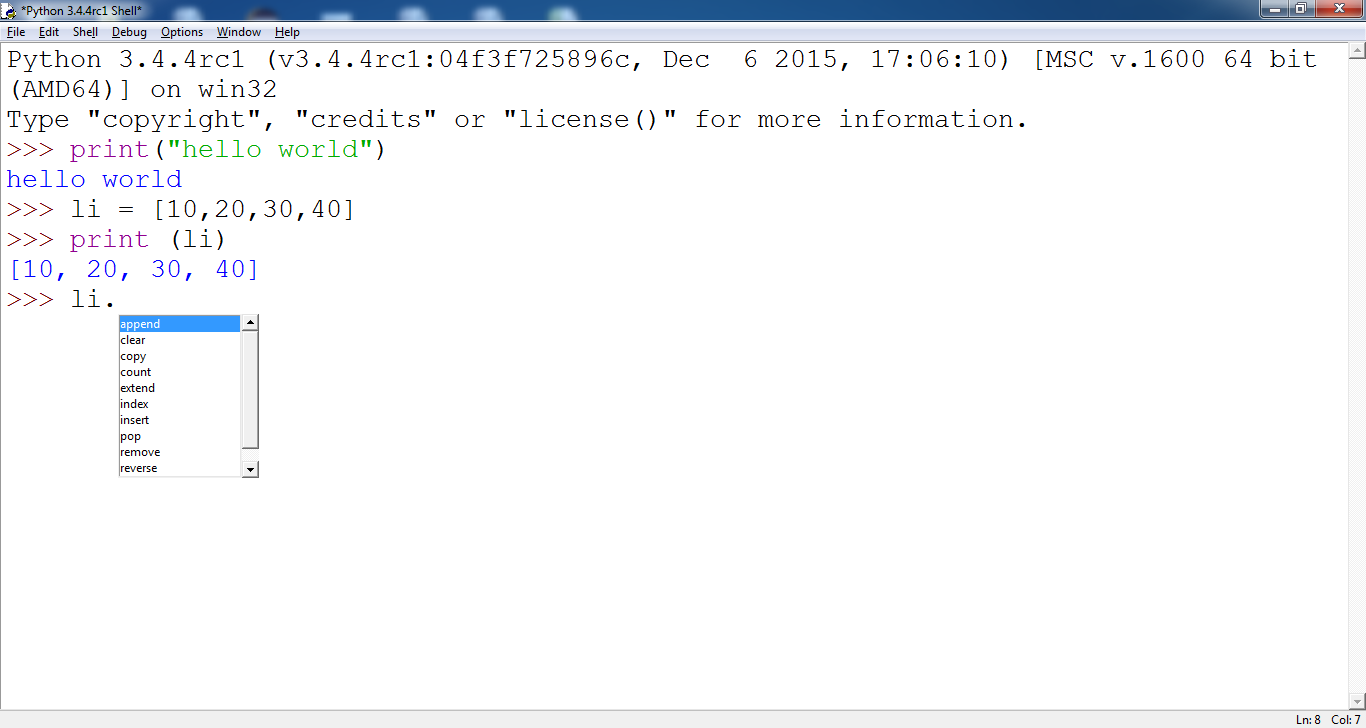
**select “Python IDLE 3.4 GUI “**



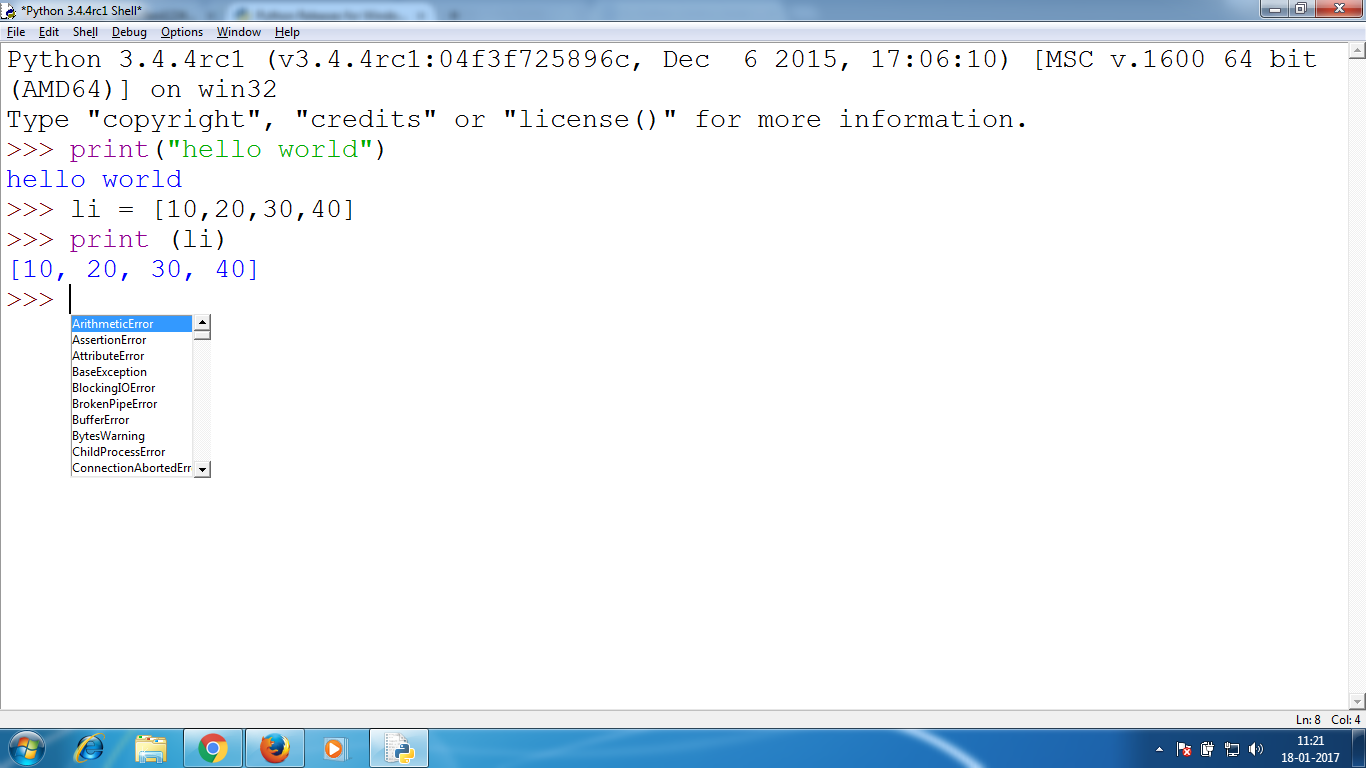
(or)



**Press (Control + Space) returns list of Methods and classes**



**Press (Control + Space)**



Expression

Expressions are just values combined with operators, and evaluate down to a single value.

Go to python Shell

>>> 2 + 2  
 4  
>>>

In Python, 2 + 2 is called an *expression,*

Expressions consist of *values* (such as2) and

*operators* (such as +), and they can always *evaluate* (that is, reduce) down to a **single value**.

>>> print(' Hello World ')

Hello World

>>> 24 + 24

48

>>> 12 \* 3

36

>>> 7/3

2.3333333333333335

>>> 7//3

2

>>> 7%3

1

>>> 4 \* 3 + 7 # 40 or 19

19

>>> 4 \* (3 + 7)

40

>>>

**Math Operators from Highest to Lowest Precedence**

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Operation** | **Example** | **Result** |
| \*\* | Exponent | 2 \*\* 3 | 8 |
| % | Modulus/**remainder** | 22 % 8 | 6 |
| // | Integer division/**floored quotient** | 22 // 8 | 2 |
| / | Division | 22 / 8 | 2.75 |
| \* | Multiplication | 3 \* 5 | 15 |
| - | Subtraction | 5 - 2 | 3 |
| + | Addition | 2 + 2 | 4 |

* **The order of operations** (also called *precedence*) of Python math operators.
* The \*\* operator is evaluated first.
* The \*, /, //, and % operators are evaluated next, from **left to right.**
* and the + and - operators are evaluated last (also from left to right).

>>> 4/4\*4

4.0

>>> 4\*4/4

4.0

>>> 4\*5/5

4.0

>>>

>>> 8 + 3 \* 4 # 44 or 20

20

>>> 8 / 3 \* 4

10.666666666666666

>>>

>>> 6 + 4 \* 3

18

>>> (6+4) \* 3

30

>>>

>>> (6+4) \*3

30

>>> 17 / 3 \* 4 # Left to Right

22.666666666666668

>>>

>>> 6 + 4 \*3

18

>>>

>>> 9/2 # Division Result

4.5

>>> 9 // 2 # QUotient

4

>>> 9 % 2 # Remainder

1

>>>

>>> 9 \*\* 9 # Exponent

387420489

>>> 9 \*\* 999

194207916858072401073330513240517841169895831937243168645765334645631807358586165476831829984964567897289883410682808509863485381763945405279379355788182053541434708898886353264614403164257835946591015853500491562156765579388944516423770646547300211711400609344237550775485394558425026601257627110879613741893863295847627378504481736441703291029360564416718984718052676789493826372811349572386149787861703350363229770343522164432121091627871310618608734044108407173015970850780786711471108639762810760748899301375323974504010469298672123113693793242558662498267897607159946316136440215024585534972601864730717278590674861331708227340510282977338127859756479389076075528672989549862138485404935127984793120586289288424045660573066638008624179879066798350622453419082976217706653276687992598885030141711458658381360884807741768071789239593772708382532520992894115725948613681993478965648216640862698897925988931145600683858128653568049999074868783790048889

>>>

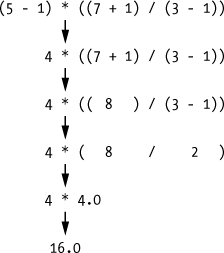
Examples ::

>>> **2 + 3 \* 6**  
 20  
>>> **(2 + 3) \* 6**  
 30  
>>> **48565878 \* 578453**  
 28093077826734  
>>> **2 \*\* 8**  
 256  
>>> **23 /** 7  
 3.2857142857142856  
>>> **23 // 7**  
 3  
>>> **23 % 7**  
 2  
>>> **2 + 2**  
 4  
>>> **(5 - 1) \* ((7 + 1) / (3 - 1))**  
 16.0

**>>> 7 // 2 \* 8**

**24**

**Evaluating an expression reduces it to a single value**



If you type bad Python instruction,

Python won’t be able to understand it and will display a **SyntaxError error message**,

>>> 5+

SyntaxError: invalid syntax

>>> 42+5+\*2

SyntaxError: invalid syntax

**DATA Type**

* A ***data type*** is a **category of values**, and every value belongs to exactly one data type.
* The values -2 and 30, for example, are said to be ***integer*** values.
* Numbers with a decimal point, such as 3.14, are called ***floating-point number****s* (or *floats*).
* The value 42 is an integer, the value 42.0 would be a floating-point number.

**The most common data types in Python are:**

**Data type Examples**

**Integers**  -2, -1, 0, 1, 2, 3, 4, 5

**Floating-point numbers** -1.25, -1.0, -0.5, 0.0, 0.5, 1.0, 1.25

**Strings**  'a', 'aa', 'aaa', **'Hello!\*\*',** **'145\*c)&&\*ats'**

* Python programs can also have text values called ***strings*.**
* Always surround your string in **single quote (')** characters (as in 'Hello' or 'Goodbye cruel world!')

Integer + and \*

>>> 2 + 3

5

>>> 4 + 5.5

9.5

>>> 'hello' + 'world'

'helloworld'

>>>

>>> 2 + 3 # Integer + Integer :: Integer

5

>>> 2 + 3.5 # Integer + float :: float

5.5

**>>> '2' + '222'**  # String + string :: String

'2222'

>>>

**String Concatenation**

>>> **'Hello' + 'world'**

'Helloworld'

**>>> 'hello'+10**

Traceback (most recent call last):

File "<pyshell#16>", line 1, in <module>

'hello'+10

TypeError: Can't convert 'int' object to str implicitly

**String Replica**

>>> **'Hello' \* 10**

'HelloHelloHelloHelloHelloHelloHelloHelloHelloHello'

>>>

>>> 'hello' + 10

Traceback (most recent call last):

File "<pyshell#0>", line 1, in <module>

'hello' + 10

TypeError: must be str, not int

>>> 'hello' + '10'

'hello10'

>>>

**string + String :::: String Concatenation**

>>> **'python program '** **+** ' language\* 123 first @#@ '

'python program language\* 123 first @#@ '

>>> 2 + 3

5

>>> 2 \* 3

6

>>> '2'+3

Traceback (most recent call last):

File "<pyshell#14>", line 1, in <module>

'2'+3

TypeError: Can't convert 'int' object to str implicitly

>>> '2'\*3

'222'

>>>

**string \* string :::: Error**

>>> 'python program ' \* ' language\* 123 first @#@ '

Traceback (most recent call last):

File "<pyshell#19>", line 1, in <module>

'python program ' \* ' language\* 123 first @#@ '

TypeError: can't multiply sequence by non-int of type 'str'\

**string \* 3 :::: String Replica**

>>> ' python prog ' \* 3

' python prog python prog python prog '

# >>> 24 + 24 # integer + integer ::: Integer

# 48

# >>> '24' + '24' # string + string ::: string Concatenation

# '2424'

# >>>

# 

# 

# String Concatenation and Replication

* The meaning of an operator may change based on the data types of the values next to it.
* For example, + is the addition operator when it operates on **two integers or floating-point values**.
* when **+ is used on two string values**, it joins the strings as the ***string concatenation*** operator.

>>> '24'+'24'

'2424'

>>> 'abc24' + 'abc'

'abc24abc'

>>> 24 \* 2

48

>>> **'24'\*2**

'2424'

>>> '24' \* 10 # string \* integer ::: string Replica

'24242424242424242424'

>>> '24' + ' 24' # String + string ::: string Concatenation

'24 24'

>>>

>>> **'24'+10**

Traceback (most recent call last):

File "<pyshell#7>", line 1, in <module>

'24'+10

TypeError: Can't convert 'int' object to str implicitly

>>>

>>> '24'\*'24'

Traceback (most recent call last):

File "<pyshell#21>", line 1, in <module>

'24'\*'24'

TypeError: can't multiply sequence by non-int of type 'str'

>>>

The **+ operator** on a **string and an integer value**,

>>> ' prem' \* 10

' prem prem prem prem prem prem prem prem prem prem'

The \* operator is used for multiplication when it operates on two integer or floating-point values.

But when the \* operator is used on one string value and one integer value, it becomes the ***string replication* operator.**

Python will not know how to handle this, and it will display an error message.

The error message Can't convert 'int' object to str

>>> 'prem' + 42

Traceback (most recent call last):

File "<pyshell#20>", line 1, in <module>

'prem' + 42

TypeError: Can't convert 'int' object to str implicitly

>>>

**Expressions : Evaluate to Single value**

**Operators** (+, -, \*, /, //, %, and \*\* for **math operations**, and + and \* for **string operations**)

**Data types** (integers, floating-point numbers, and strings)

>>> 9/2

4.5

>>> 9//2

4

>>> 9%2

1

>>>

# Variables

* A ***variable*** is like a box in the **computer’s memory** where you can **store a single value**.

## Assignment Statements

* To store values in variables use an ***assignment statement***.
* An assignment statement consists of a variable name, an equal sign (called the *assignment operator*), and the value to be stored.
* If you enter the assignment statement **a = 24,** then a variable named spam will have the integer value 42 stored in it, When a new value is assigned to a variable, **the old one is forgotten**

>>> i = 24

>>> i

24

>>> i+24

48

>>> i

24

**>>> i**

**48**

>>>

>>> a = 24

>>> b = 6

>>> c = a + **B**  **# Python CASE Sensitive**

Traceback (most recent call last):

File "<pyshell#2>", line 1, in <module>

c = a + B

NameError: name 'B' is not defined

>>> c = a + b

>>> c

30

>>> c\*30

900

**>>> c**

**30**

>>> c = c\*30

>>> c

900

>>>

* A variable is ***initialized*** (or created) the first time a value is stored in it.
* After that, you can use it in **expressions** with other variables and values.
* When a variable is assigned a new value, the old value is forgotten

## 

## 

## Variable Names Rules

You can name a variable anything as long as it obeys the following three rules:

1. It can be only one word.
2. It can use only **letters, numbers, and the underscore** (\_) character.
3. It can’t begin with a number.

**Variable names are case-sensitive, meaning that spam, SPAM, Spam, and sPaM are four different variables**

>>> empid=123

>>> emp id =123 **# Space not allowed**

SyntaxError: invalid syntax

>>> emp-id=123 **# - special character - not allowed**

SyntaxError: can't assign to operator

>>> emp\_id=123 **# \_ accepted (underscore)**

>>> 9empid =123 **# can’t start Numerics**

SyntaxError: invalid syntax

>>> empid\* =123 **# can’t contain special character**

SyntaxError: invalid syntax

>>> \_empid = 123

**a is variable of type string**

**input() default accepts String type**

>>> a = input('Enter value for a ')

Enter value for a 123

>>> a

'123'

>>> a \* 10

'123123123123123123123123123123'

>>>

>>>

>>>

>>>

>>>

>>> a + 10

Traceback (most recent call last):

File "<pyshell#110>", line 1, in <module>

a + 10

TypeError:

**int() :: Converts String to Integer**

>>> a = **int**(input('Enter value for a '))

Enter value for a 123

>>> a

123

>>> a + 10

133

>>> a \* 10

1230

>>>

>>> a = **input('Enter value for a ')**

Enter value for a 45

**>>> a**

**'45'**

>>> a + 10

Traceback (most recent call last):

File "<pyshell#54>", line 1, in <module>

a + 10

TypeError: Can't convert 'int' object to str implicitly

**>>> a \* 10**

'45454545454545454545'

>>>

a which type :

>>> type(a)

<class 'str'>

>>>

**Accepting “a” as integer format**

>>> a = **int(input('Enter value for a '))**

Enter value for a 24

>>> a

24

>>> a + 10

34

>>> a \* 10

240

>>> type(a)

<class 'int'>

>>>

**Keywords can’t use as Variable Names**

>>> if = 24

SyntaxError: invalid syntax

>>> for = 45

SyntaxError: invalid syntax

>>>

## 29 keywords

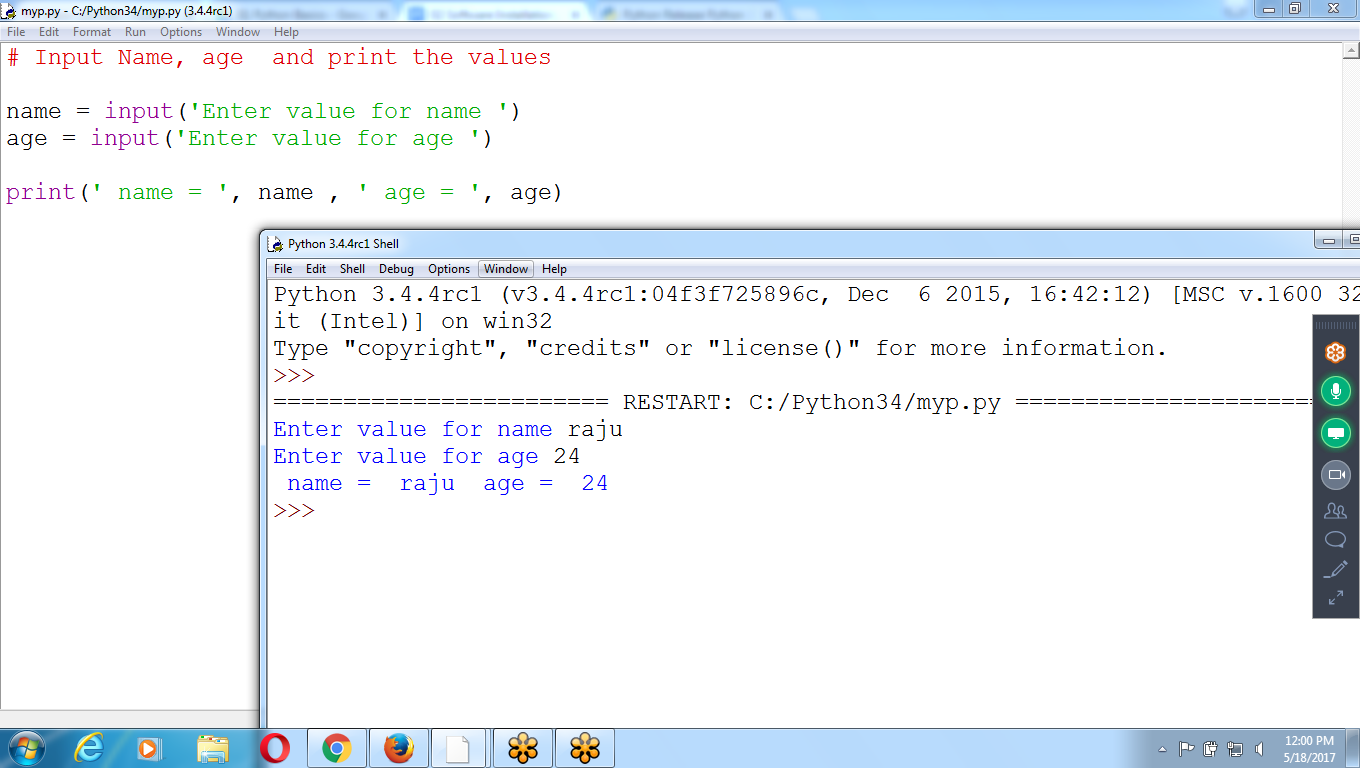
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| and | def | exec | if | not | return |
| assert | del | finally | import | or | try |
| break | elif | for | in | pass | while |
| class | else | from | is | print | yelid |
| continue | except | global | lambda | raise |  |

These are the words which are reserved for python and we can't use them for our variables.

* The **interactive shell** is good for running Python instructions **one at a time**,
* To write entire Python programs, type the instructions into the **file editor**.
* To open the file editor in **IDLE select File▸New File**
* To Save File : select **File▸Save As.**
* To **RUN** program. Select **Run**▸**Run Module** or just press the **F5** key.
* Have to press **F5** from the **file editor window,** not the interactive shell window.
* To reload a saved program, select **File▸Open** from the menu

**First Program**

**File Menu - NEW :**

****

**# Input Name, age and print the values**

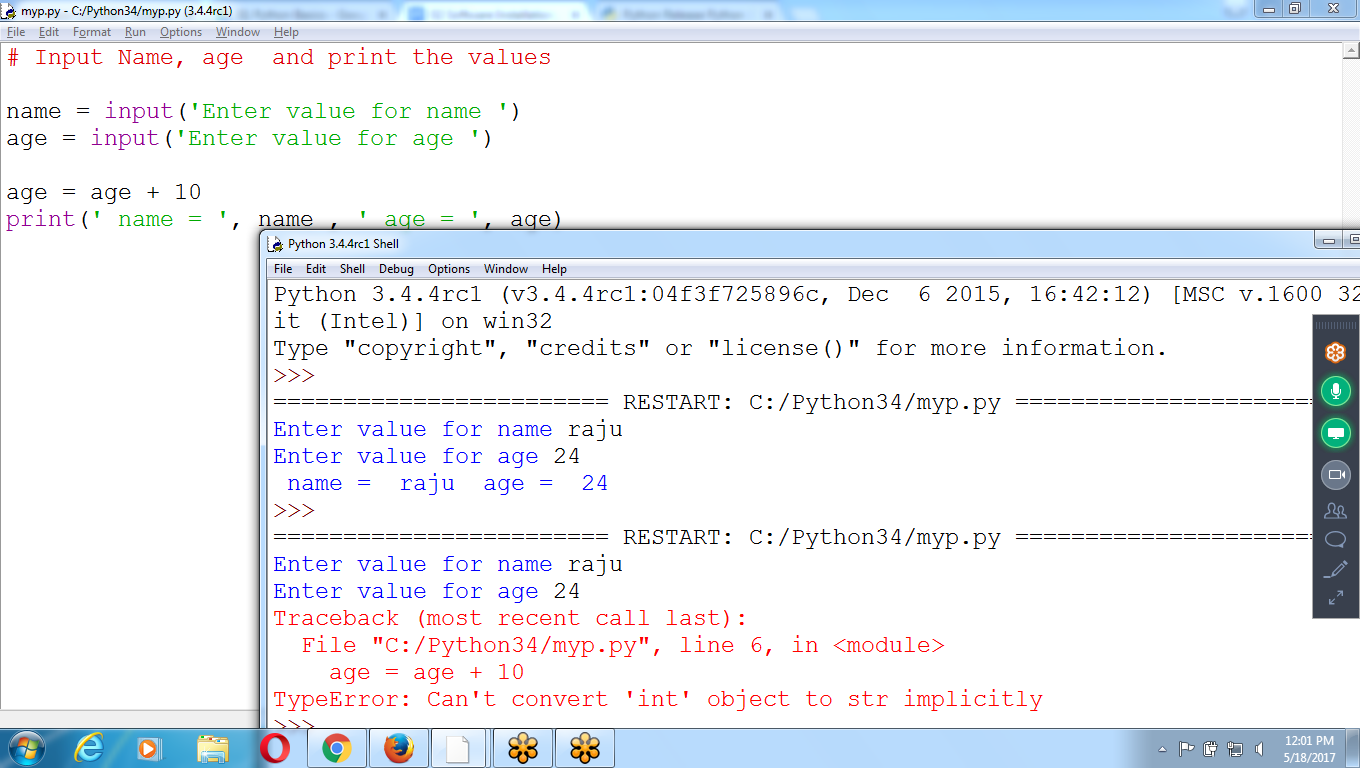
**name = input('Enter value for name ')**

**age = input('Enter value for age ')**

**print(' name = ', name , ' age = ', age)**

**Addition on age value :**

**Result : Error ( string + integer is an Error )**

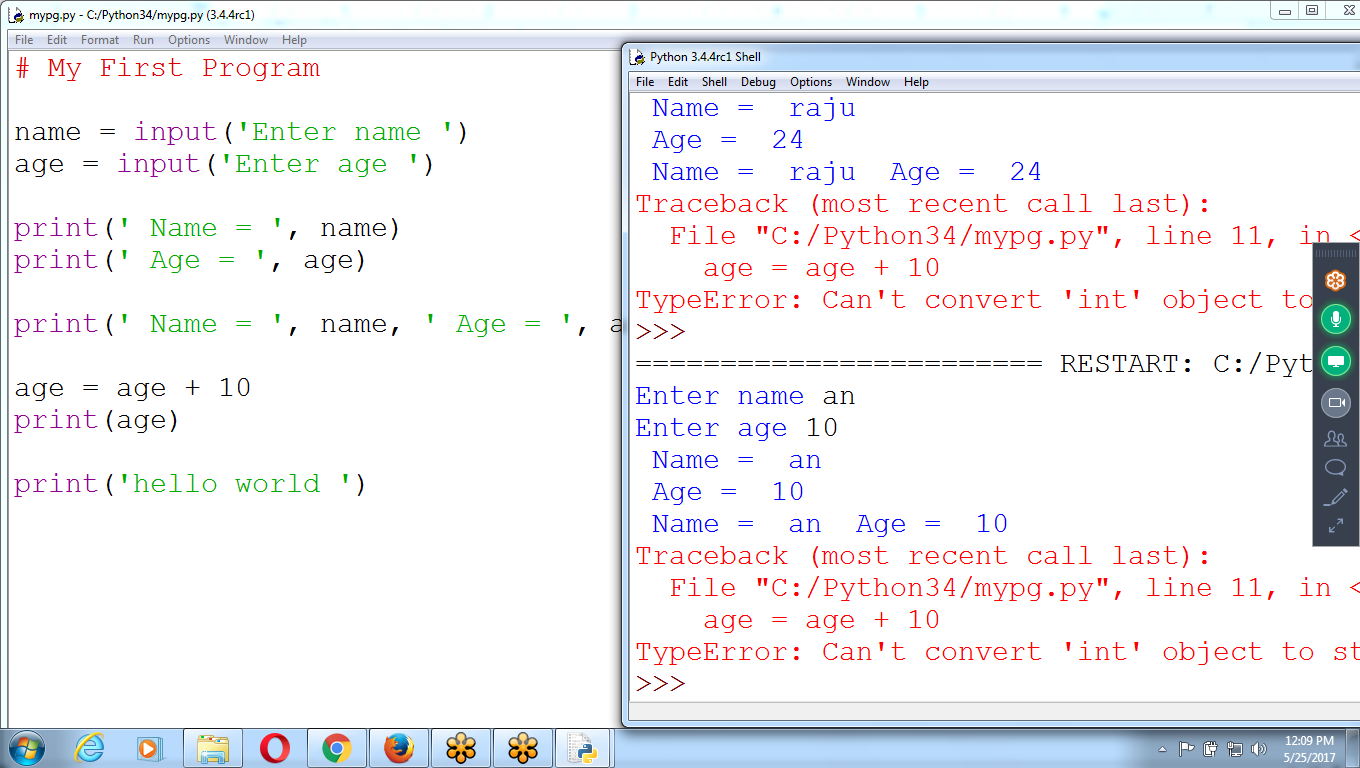
****

**age = age +10**

**age as string type; string + 10 ::: Error**

**Python follows Interpreter Translation, it is not a COMPILER**

**Line by Line Translation**

****

**Compiler Interpeter**

**1000 Line program (Java, C#, cobol) (Python, Basic)**

**Waiting Time :: MORE LEss**

**Turnaround time: LEss More**

**Java follows : BOTH (Compiler and Interpreter)**

**writing program and compiling before execution: Compilation**

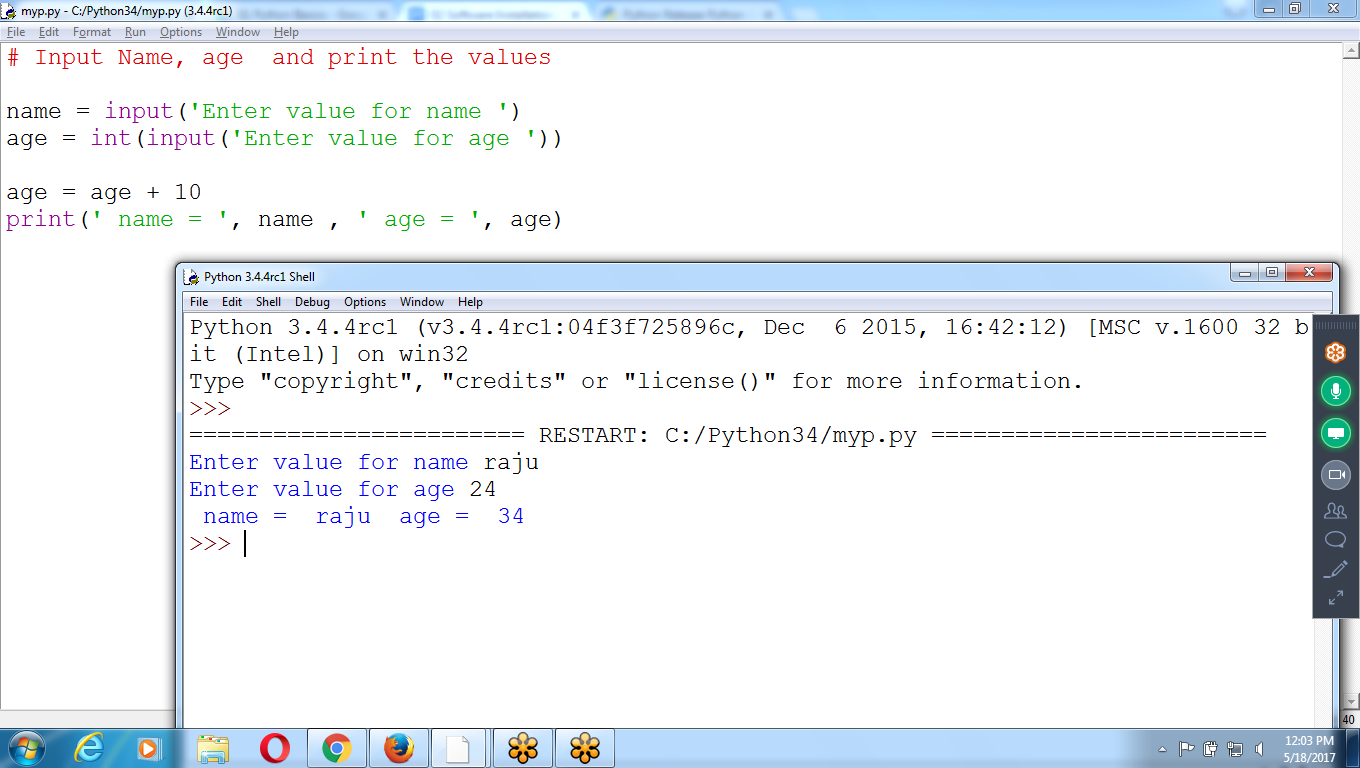
**At the time of Execution : Interpreter**

**Python Follows : Interpreter**

**Default : input() takes string type**

**int() : converts string to int type**

**now age is converted from string to int type**

****

**# Input Name, age and print the values**

**name = input('Enter value for name ')**

**age = int(input('Enter value for age '))**

**age = age + 10**

**print(' name = ', name , ' age = ', age)**

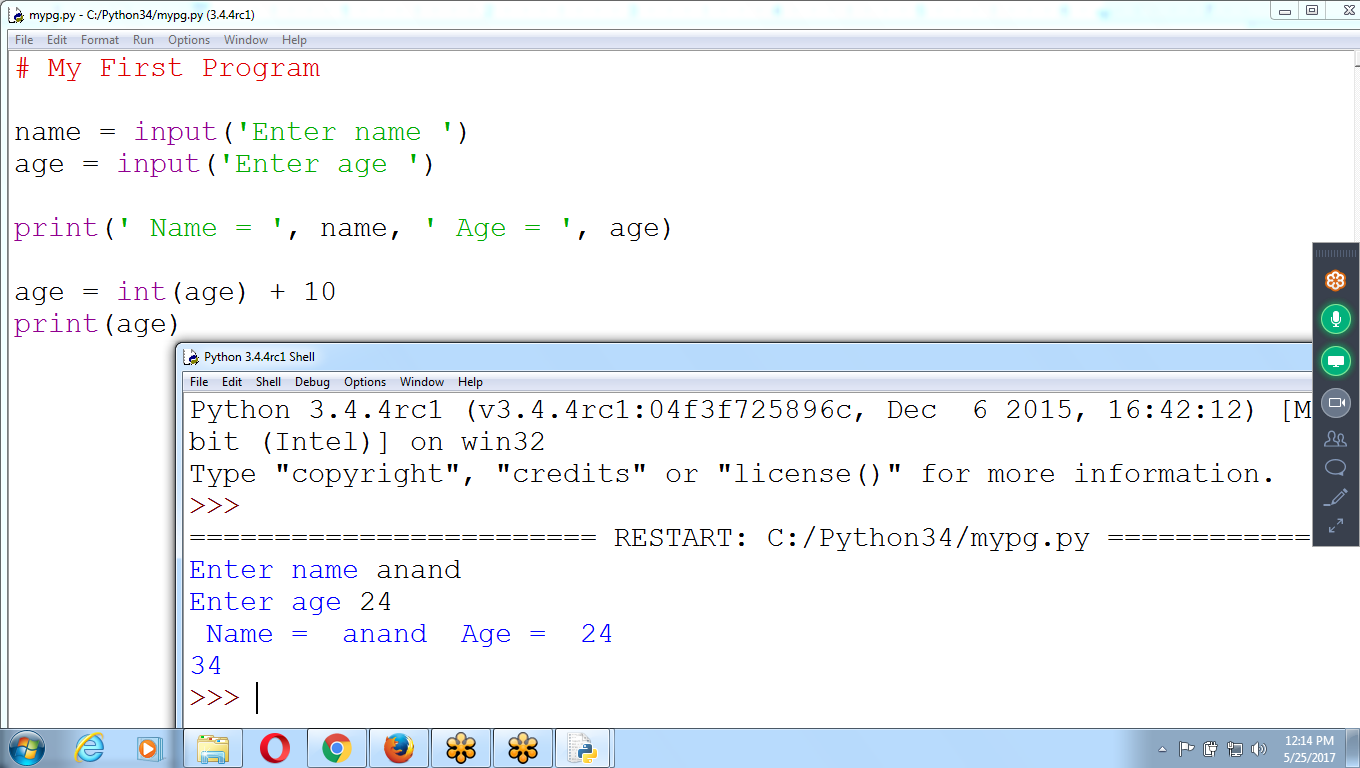
**Testing: input as integer using int()**

**age = int(input(‘enter age ’)**

**age is string**

**int(age) : Converting string to int for that statement**

**age = int(age) + 10 # but internally age is String**

****

**# Input Name, age and print the values**

**name = input('Enter value for name ')**

**age = input('Enter value for age ') # age is like STRING Type**

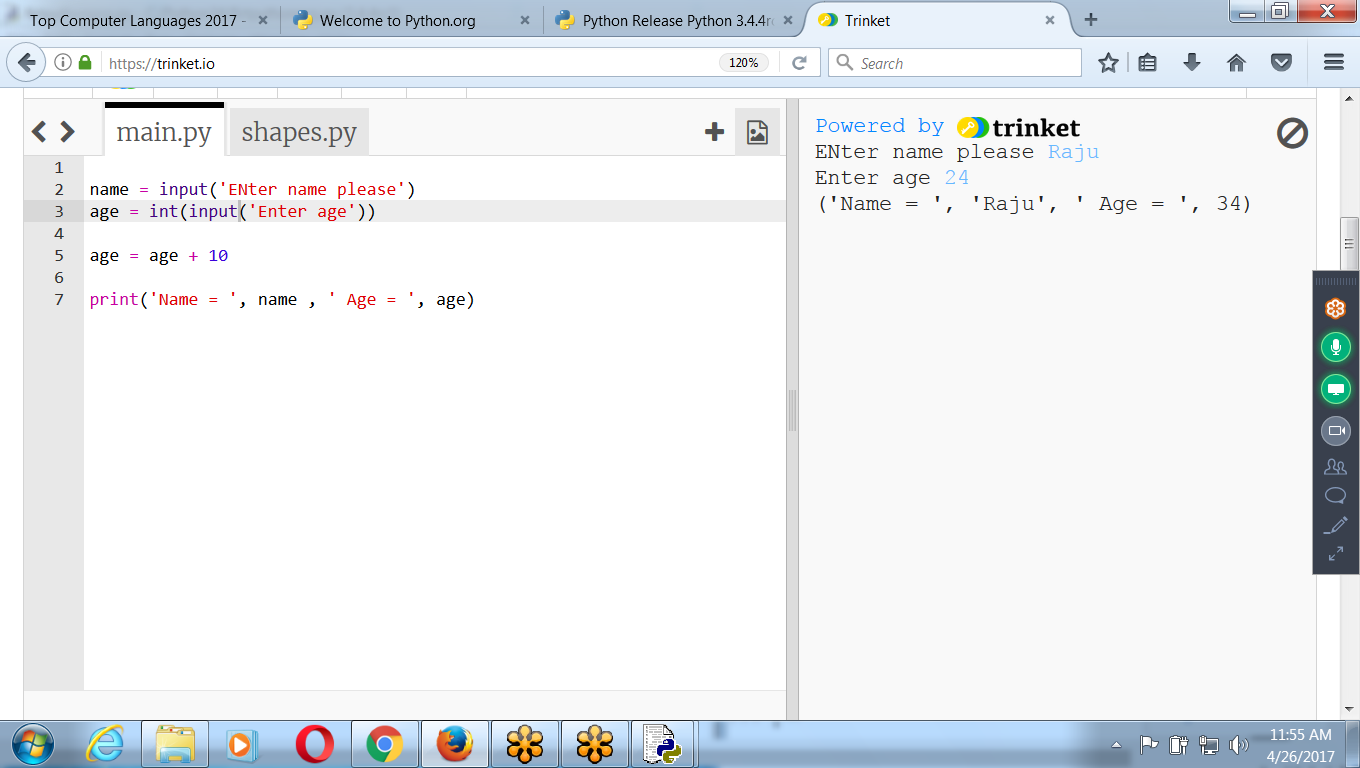
**#age = age + 10 # string + 10 is an Error**

**age = age \* 10 # Working as String Replica**

**print(' name = ', name , ' age = ', age)**

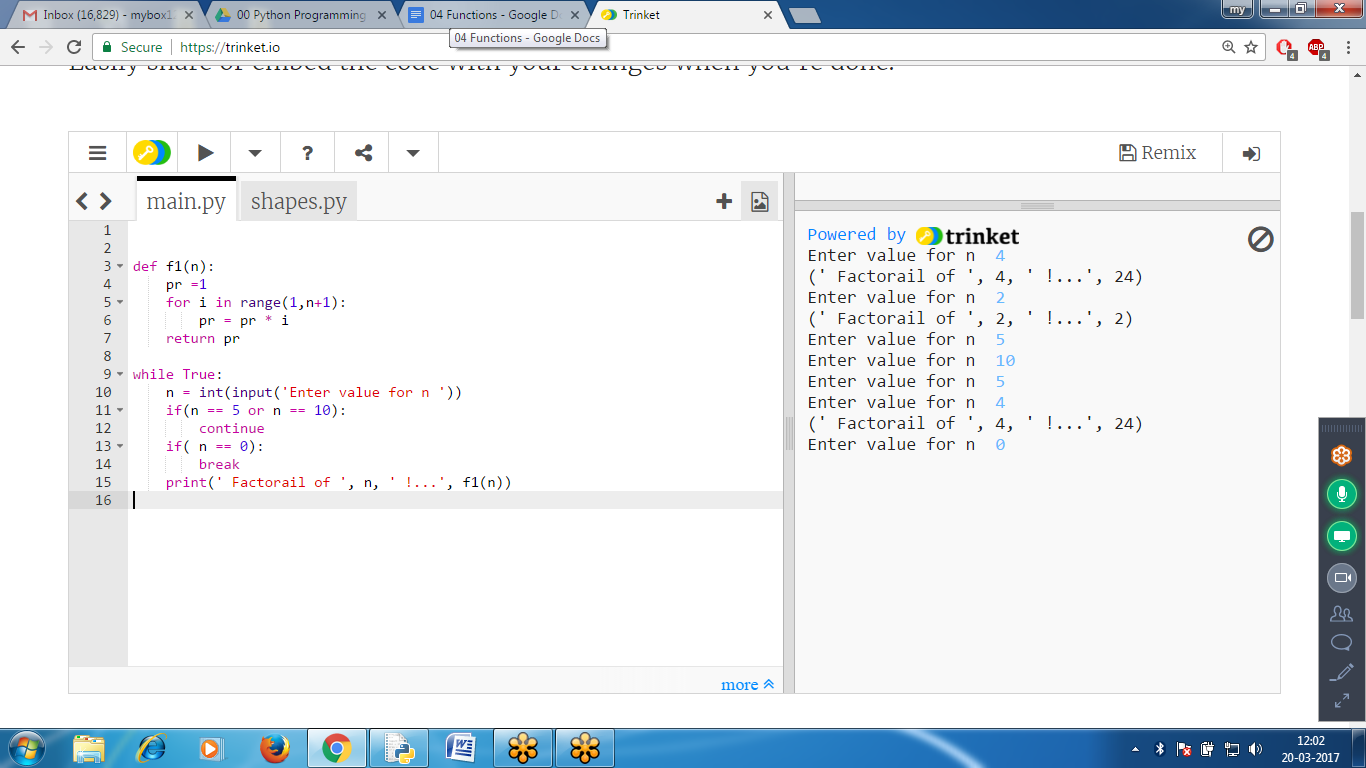
**Can RUn Python programs BROWSER based, without PYTHON software**

**https://trinket.io/**

****

In Trinket Run programs (Web based)

<https://trinket.io/>



**Python Basics : 10 am**

**1 Installation, Expression, Data types, Variables,**

**Execution and Operators**

**2 Operators and Flow Controls (Decision) (If, if-else)**

**3 Loops for, while, Range**

**4 Functions and Collections( List, tuple, Dictionary)**

**PYTHON (30 Hours) : 9.30 to 10.30 am from Thursday**

**R Programing: 10.30 to 11.30 am**

**Python : Strings**

**Python : Regular Expressions**

**Python : Files**

**Python: Debugging, Exceptions**

**Python : Generators and Decorators**

**Python : Functions**

**Python : Collections (List, set , Dictionary)**

**python : OOPS**

**Advance Python ( 30 Hours)**

**Web, GUI, Database, Data Structures**

**Gaming, Multithreading, Numpy, scipy**

**Networking, Email, Scraping**

**Jupyter, Pycharm (EDITORS)**

**Python Developer :**

**Python (30 H) +**

**Advanced Python (30) + Python Projects**

**dJango (30 Hours) + Web Projects**

**Other Technologies :**

**Python (30 Hours) +**

**Data Science or AWS or**

**Devops or Big data or**

**Selenium or Open stack**

**Data Science : 3 Months (250 Hours)**

**Mon to Fri : 9.30 to 11.30**

**sat and sun :**

**Python + R Lang + Machine Learning ( 2 months)**

**Deep Learning + NLP + Predictive Analysis +**

**Artificial Intelligence ( 2 Months)**