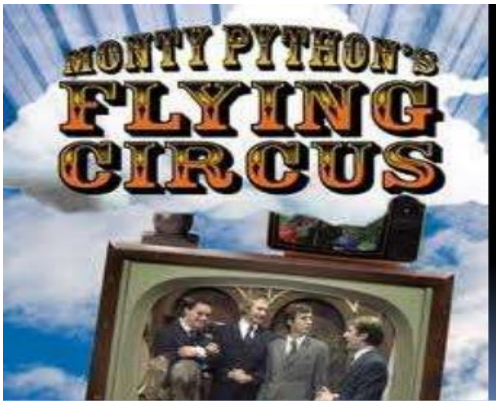
PYTHON PROGRAMMING

UNIT – I:
Introduction:

History of Python :

The History of Python Starts with ABC. ABC (All Basic Codes) is a General-Purpose Programming Language Developed in the Netherlands. The greatest achievement of ABC was to influence the design of python

- Python laid its foundation in the late 1980s.
- The implementation of Python was started in the December 1989 by **Guido Van Rossum** at CWI (***Centrum Wiskunde & Informatica)*** -> ("National Research Institute for Mathematics and Computer Science") in Netherland.
- In February 1991, van Rossum published the code (labeled version 0.9.0) to alt.sources.
- In 1994, Python 1.0 was released with new features like: lambda, map, filter, and reduce.
- Python 2.0 added new features like: list comprehensions, garbage collection system.
- On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify fundamental flaw of the language.
- Python is influenced by following programming languages:
  - ABC language.
  - Modula-3



what about the name "Python": Most people think about snakes, and even the logo depicts two snakes, but the origin of the name has its root in British comic. **Guido Van Rossum** is a big fan of python's flying circus

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Need / Features of Python Programming Language

1. Easy to Learn and Use:

Python is easy to learn and use. It is developer-friendly and high level programming language.

Python is very easy to code. Compared to other popular languages like Java and C++, it is easier to code in Python. Anyone can learn python syntax

Mastering Python requires learning about all its advanced concepts and packages and modules. That takes time.

2. Free and Open-Source:

Firstly, Python is freely available. You can download it from the following link

https://www.python.org/downloads/

Secondly, it is open-source. This means that its source code is available to the public. You can download it, change it, use it, and distribute it. This is called FLOSS(Free/Libre and Open Source Software). As the Python community, we're all headed toward one goal- an ever-bettering Python.

3. high-level Language:

When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, we don't need to remember the system architecture. **It looks more like a readable , human language.**

4. Portable:

Let's assume you've written a Python code for your Windows machine. Now, if you want to run it on a Mac, you don't need to make changes to it for the same. In other words, you can take one code and run it on any machine, there is no need to write different code for different machines. This makes Python a portable language.

5. Interpreted:

If you're any familiar with languages like C++ or Java, you must first compile it, and then run it. But in Python, there is no need to compile it. Internally, its source code is converted into an immediate form called byte code. So, all you need to do is to run your Python code without worrying about linking to libraries.

By interpreted, mean that the source code is executed line by line, and not all at once. Because of this, it is easier to debug your code.

6. Object-Oriented:

A programming language that can model the real world is said to be object-oriented. It focuses on objects, and combines data and functions.

Python supports procedure-oriented programming as well as object-oriented programming. In *procedure-oriented* languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In *object-oriented* languages, the program is built around objects which combine data and functionality.

7. Extensible:

you can write some of your Python code in other languages like c or C++. If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++ and then use them from your Python program.

8. Embeddable:

we can put code in other languages in our Python source code. However, it is also possible to put our Python source code in a different language like C/C++.

9. Large Standard Library:

There are over 300 standard library modules which contain classes for a wide variety of programming tasks that you can use so you don't have to write your own code for every single thing. There are libraries for regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and a lot of other functions for rapid application development.

10. Python is Dynamically Typed:

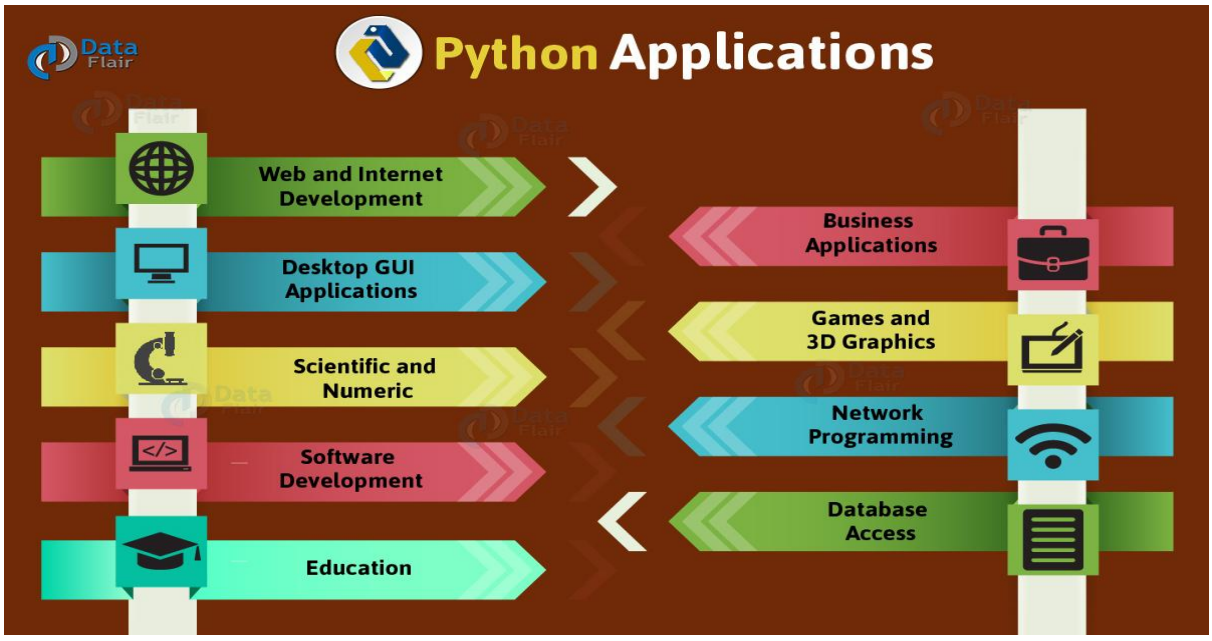
Python is dynamically-typed. This means that the type for a value is decided at runtime, not in advance. This is why we don't need to specify the type of data while declaring it.

11. **GUI Programming**: (Graphical user interface) You can use Tk to create basic GUIs.

12. **Important: Python has been an important part of Google since the beginning and remains so as the system grows & evolves , Today dozens of Google engineers use Python Python may become popular for the** Internet **, as new platforms such as Raspberry Pi are based on it**

Applications of Python Programming

Python is freely available for Windows, Mac OS, and Linux its popularity is constantly increased. Python is a high-level general purpose programming language it is used to develop a wide range of applications.



1) **Web and Internet Development**:
   We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, beautifulSoup, Feed parser etc. It also provides Frameworks such as Django, Pyramid, Flask etc to design and develop web based applications. Some important developments are: Python Wiki Engines, Python Blog Software etc.

- Frameworks such as Django and Pyramid.
- Micro-frameworks such as Flask and Bottle.
- Advanced content management systems such as Plone and django CMS.

Python's standard library supports many Internet protocols **:**

- HTML and XML
- JSON
- E-mail processing.
- Support for FTP, IMAP, and other Internet protocols.
- Easy-to-use socket interface.

2) GUI-Based Desktop Applications:

Python provides Tk GUI library to develop user interface in python based application. Python has simple syntax, modular architecture, rich text processing tools and the ability to work on multiple operating systems which make it a desirable choice for developing desktop-based applications. There are various GUI toolkits like wxPython, PyQt or PyGtk available which help developers create highly functional Graphical User Interface (GUI). The **Kivy** is popular for writing multi touch applications.

3) Scientific and Numeric Applications:

Python is popular and widely used in scientific and numeric computing. Some useful library and package are SciPy, Pandas, IPython, NumPy etc. SciPy is group of packages of engineering, science and mathematics.

- SciPy – A collection of packages for mathematics, science, and engineering.
- Pandas--A data-analysis and -modeling library
- IPython – A powerful shell for easy editing and recording of work sessions. It also supports visualizations and parallel computing.
- NumPy lets us deal with complex numerical calculations.

4) Software Development Application :

Python is helpful for software development process. It works as a support language and can be used for build control, management and testing etc.

- SCons for build control.
- Buildbot and Apache Gump for automated continuous compilation and testing.
- Roundup or Trac for bug tracking and project management.

5) Python Applications in Education :

The Education System is totally turning into digital. Requirements of various tools in the learning, teaching, and managing with the help software is increased. Therefore, Python programming language becomes a front-runner in developing various tools for the modern education system. in education has huge scope as it is a great language to teach in schools or even learn on your own.

6) Python Applications in Business :

Python is also a great choice to develop ERP and e-commerce systems:

- Tryton – A three-tier, high-level general-purpose application platform.
- Odoo – A management software with a range of business applications. With that, it's an all-rounder and forms a complete suite of enterprise-management applications in-effect

7) Games and 3D Graphics :

Python has various modules, libraries and platforms that support development of games. For example, PySoy is a 3D game engine supporting Python 3, and PyGame provides functionality and a library for game development. games built using Python including Civilization-IV, Disney's Toontown Online, Vega Strike etc.

8) Network Programming :

Python plays an essential role in network programming. The standard library of Python has full support for network protocols, encoding and decoding of data and other networking concepts and it is simpler to write network programs in Python than that of C++.

There are two levels of network service access in Python. These are:

- Low-Level Access
- High-Level Access

In the first case, programmers can use and access the basic socket support for the operating system using Python's libraries, and programmers can implement both connection-less and connection-oriented protocols for programming.

Sockets are the endpoints of a bidirectional communications channel.

9) Database Access :

This is one of the hottest Python Applications. With Python, you have:

- Custom and ODBC interfaces to MySQL, Oracle, PostgreSQL, MS SQL Server, and others. These are freely available for download.
- Object databases like **Durus** and **ZODB**

✓ *Durus* is a simpler object database

✓ The Zope Object Database (*ZODB*) is an object-oriented database for transparently and persistently storing *Python* objects

10) Operating Systems :

Python is often an integral part of Linux distributions. For instance, Ubuntu's Ubiquity Installer, and Fedora's and Red Hat Enterprise Linux's Anaconda Installer are written in Python

11) Prototyping :

Besides being quick and easy to learn, Python also has the open source advantage of being free with the support of a large community. This makes it the preferred choice for prototype development. Further, the agility, extensibility and scalability and ease of refactoring code associated with Python allow faster development from initial prototype.

12) mage Processing :

Python has been used to make 2D imaging software such as Inkscape, GIMP, Paint Shop Pro and Scribus. Further, 3D animation packages, like Blender, 3ds Max

Python Programming using The REPL(shell)

A **Read–Eval–Print Loop** (**REPL**), also known as an **interactive top level** or **language shell**, is a simple, interactive computer programming environment that takes single user inputs evaluates them, and returns the result to the user;

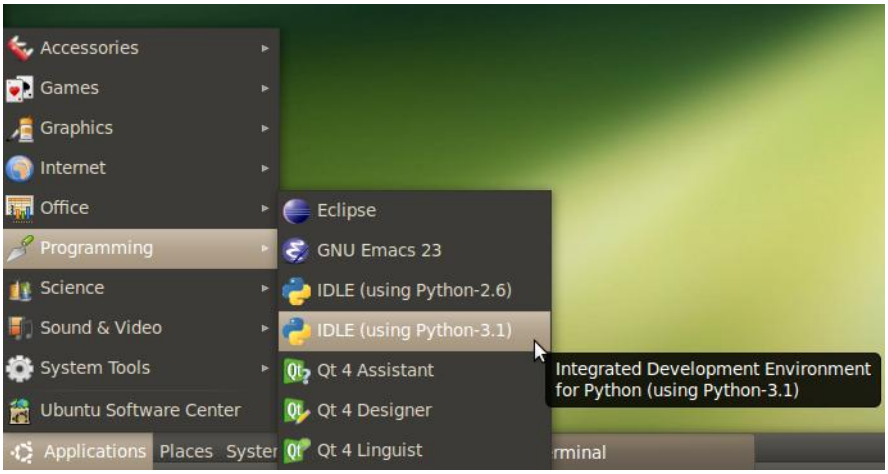
A program written in a REPL environment is executed piecewise. The term is most usually used to interactive environment. Common examples include command line shells

1. **Read:** take user input.
2. **Eval:** evaluate the input.
3. **Print:** shows the output to the user.
4. **Loop:** repeat.

IDLE startup details:

- **IDLE** short for **integrated development environment** or **integrated development and learning environment**
- **IDLE is an** development environment for Python Programming
- An IDE combines a program editor and a language environment as a convenience to the programmer.

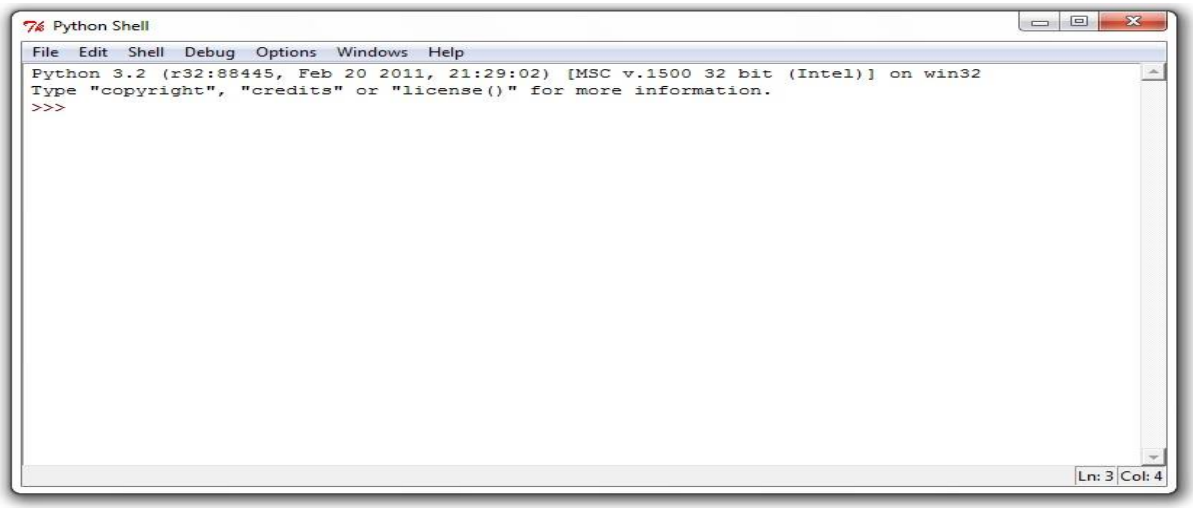
Launching IDLE:



Clicking on the IDLE selection will launch IDLE and display the Python Shell window.

Using the Python Shell Window

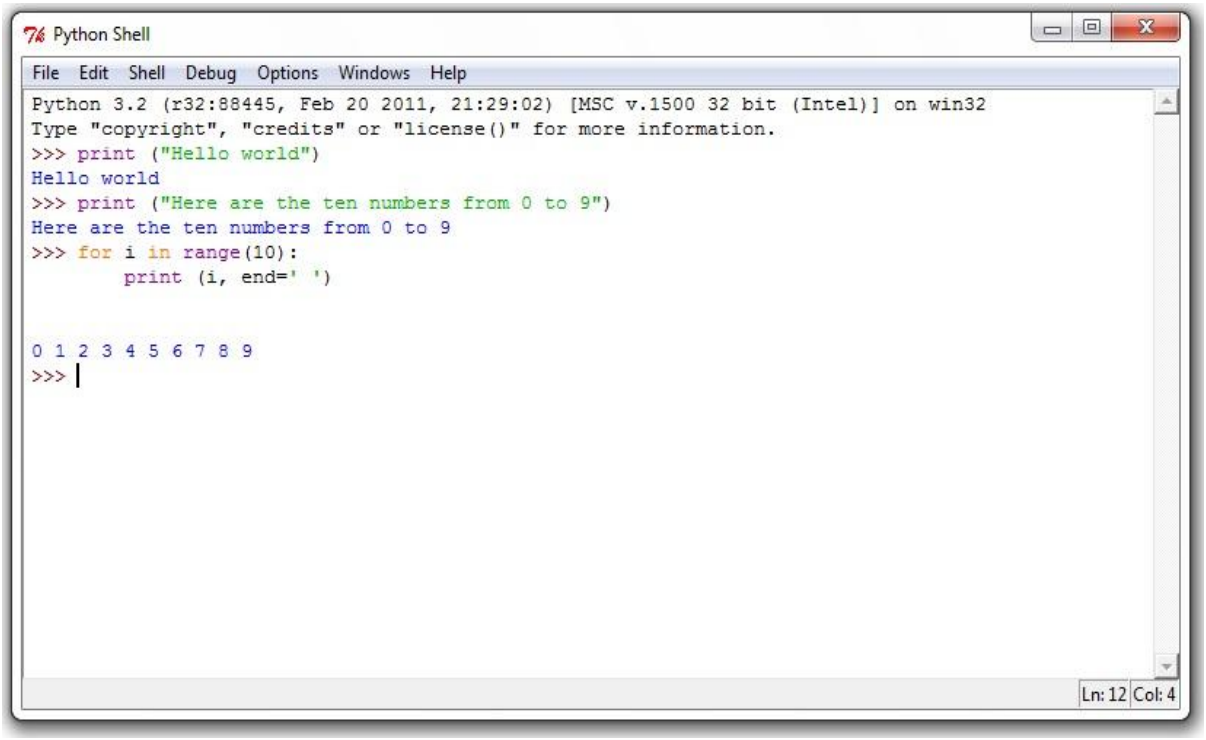
The top of IDLE's Python Shell window will look something like this:



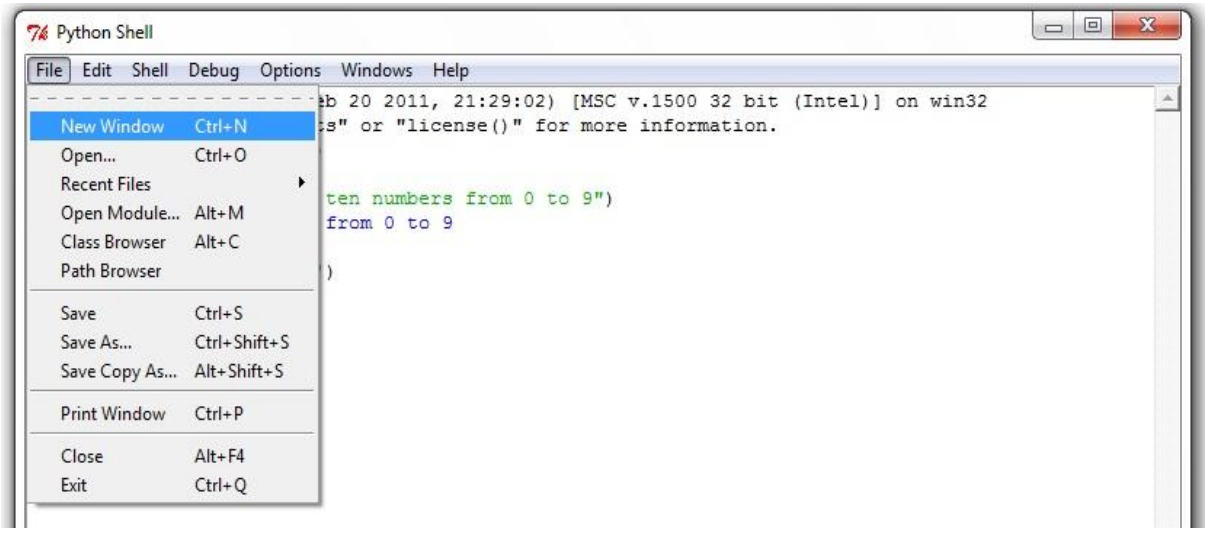
This is the main window to IDLE, and what we see right now is called the "Interpreter" (or "shell") window. The Interpreter allows us to enter commands directly into Python, and as soon as we enter in a command, Python will execute it and display its result.

'>>>' signs act as a prompt for us: Python is ready to read in a new command by giving us that visual cue. Also, we notice that as we enter in commands, Python will give us its output immediately.

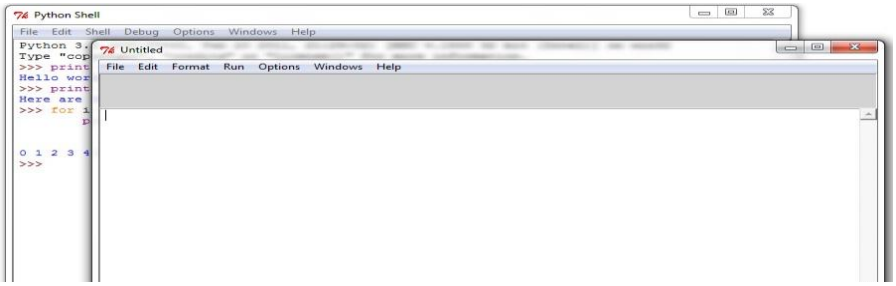
Let's try a few more commands. If we look below:



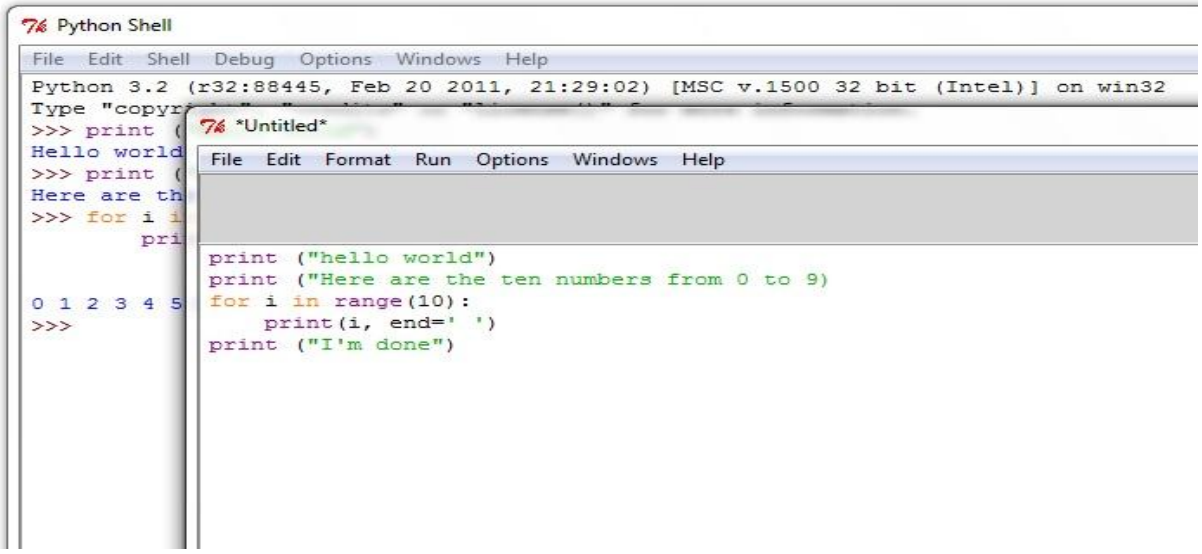
we can't directly save what's on the interpreter window, because it will include both our commands and the system's responses. let's start with opening up a new window.



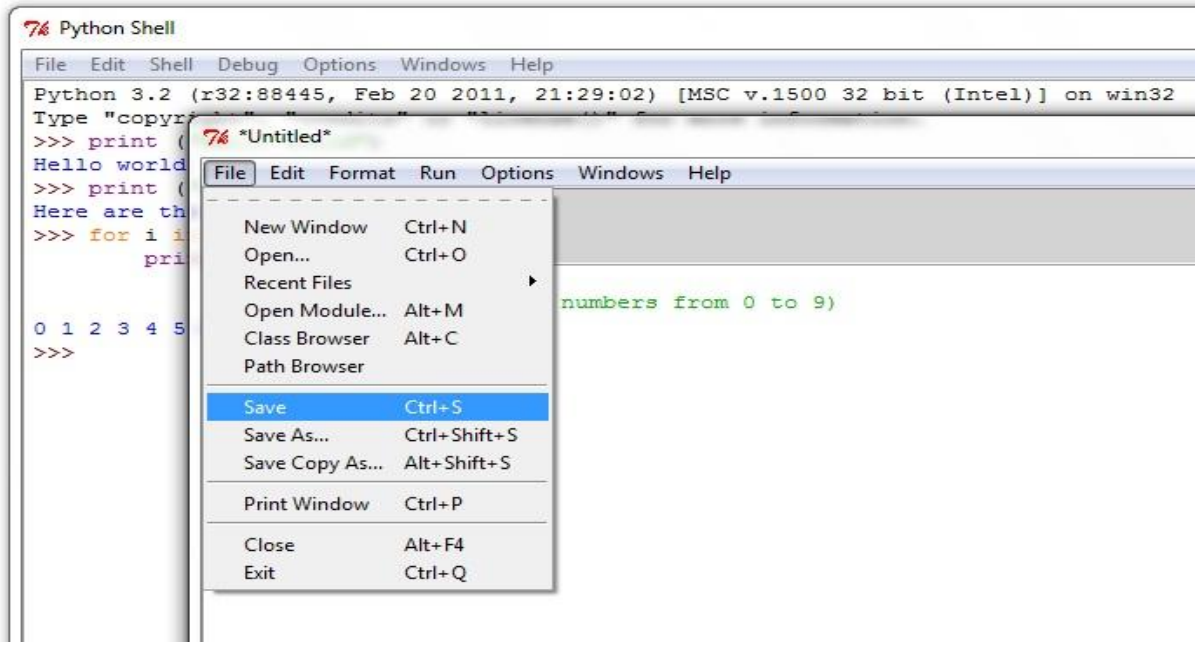
Here's the result of that New Window:



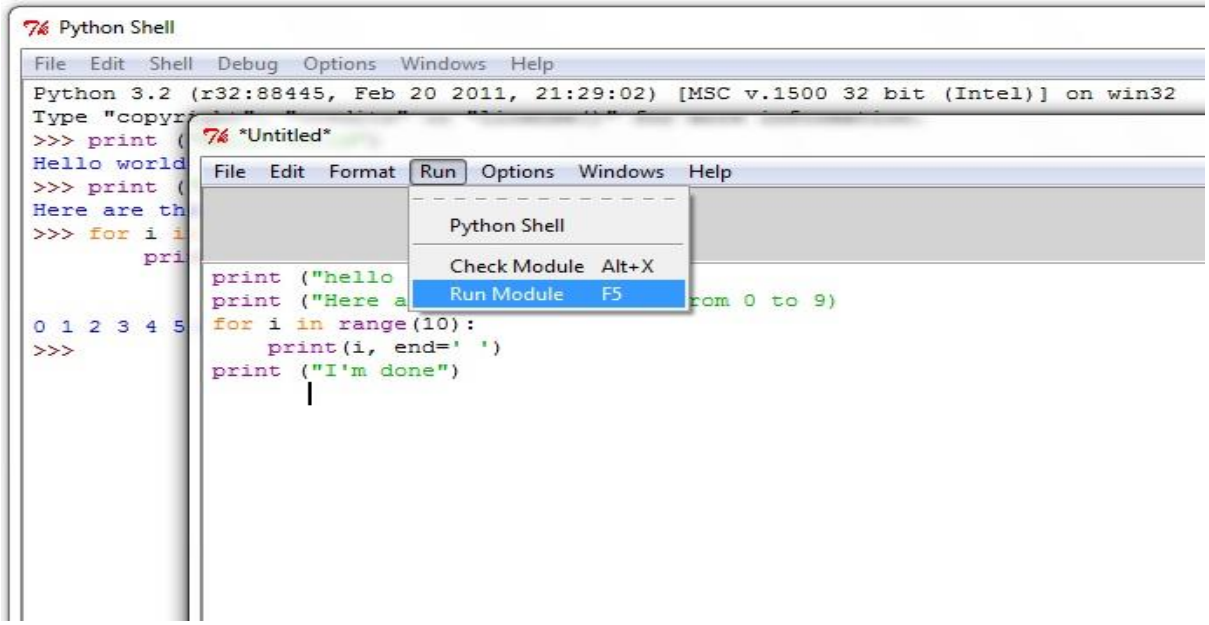
Let's do that typing those commands into our Program window.



Let's save the file now. The Save command is located under the File menu:



saved the program and run the If we look at the menus on our program window



displaying its results in the Interpreter window.

Output:

```
>>>
============================== RESTART: E:/run.py ==============================
hello world
here are the numbers from 0 to 9
0 1 2 3 4 5 6 7 8 9 iam done
>>>
```

Python variables and assignment

- Variables plays a very important role in most programming languages, and python is no exception. Variables are nothing but reserved memory locations to store values.
- This means that when you create a variable you reserve some space in memory.
- In Python, we don't need to specify the type of variable because Python is dynamically-typed. This means that the type for a value is decided at runtime, not in advance. This is why we don't need to specify the type of data while declaring it.
- Non technically, you can suppose variable as a bag to store books in it and those books can be replaced at anytime.
- Variable names can be a group of both letters and digits, but they have to begin with a letter or an underscore.

Variable Names:

A variable can have a short name (like x and y) or a more descriptive name (age, MIC, total_volume, tv9, _srikanth).

Rules for Python variables:

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)

Assignment: The equal (=) operator is used to assign value to a variable.

Example:1 Assigning values to a Variables in Python

```
distance=35
fee=66000
code="mict"
pi=3.1415926
population=100000000000
msg="Mic College of Technology"
print("distance = " + str(distance))
print("fee details = " + str(fee))
print("college code = " + str(code))
print("pi value = " + str(pi))
print("population = " + str(population))
print("message = " + msg)
```

Output:

```
 RESTART: C:/Users/Sony/AppData/Local/
distance = 35
fee details = 66000
college code = mict
pi value = 3.1415926
population = 100000000000
message = Mic College of Technology
```

Example: 2 Changing value of a variable

```
website = "sristudy.in"
# here website is a variable name
website = "mictech.ac.in"
print(website)
```

Output:

```
mictech.ac.in
```

Example: 3 Assigning multiple values to multiple variables

```
a, b, c = 5, 3.2, "MIC"
print (a)
print (b)
print (c)
```

Output:

```
5
3.2
MIC
```

Example: 4 Assign the same value to multiple variables:

```
x = y = z = "MIC CSE-A"
print (x)
print (y)
print (z)
```

Output:

```
MIC CSE-A
MIC CSE-A
MIC CSE-A
```

keywords used in Python

- Keywords are the reserved words in Python. We cannot use a keyword as variable name, function name or any other identifier.
- They are used to define the syntax and structure of the Python language.
- All keywords in Python are case sensitive. except True, False and None So, you must be careful while using them in your code.
- There are 33 keywords in Python 3.3. This number can vary slightly in upcoming versions
- to get hold the up-to-date list, you can open Python shell and run the following commands as shown in the below snippet.

```
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except',
'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return',
'try', 'while', 'with', 'yield']
```

Keyword	**Summary**	***Example***
and	Logical operator to test whether two things are both True.	<*conditional expression*> and <*conditional expression*> x>2 and x<10
as	Assign a file object to a variable. Used with with. Let your code refer to a module under a different name (also called an *alias*). Used with import.	with open(<*name of file*>,<*file mode*>) as <*object name*>: import cPickle as pickle

break	Stop execution of a loop.	for i in range(10):
		  if i%2 ==0:
		  break

class	Define a custom object.	class <*name of class*>(object):
		  “”*Your docstring*“”
		class MyClass(object):

		“”A cool function.””

continue	Skip balance of loop and begin a new iteration.	for i in range(10):
		  if i%2 ==0:
		  continue

def	Define a function.	def <*name of function*>(<argument list>):
		  “”*Your docstring*“”
		def my_function():
		  “”This does… “”

elif	Add conditional test to an if clause.	See if.
else	Add an alternative code block.	See if.
for	Create a loop which iterates through elements of a list (or other iterable).	for <*dummy variable name*> in <*sequence*>: for i in range(10):
from	Import specific functions from a module without importing the whole module.	from <*module name*> import <*name of function or object*> from random import randint
global	Make a variable global in scope. (If a variable is defined in the main section, you can change its value within a function.)	global x

if	Create a condition. If the condition is True, the associated code block is executed. Otherwise, any elif commands are processed. If there are none, or none are satisfied, execute the else block if there is one.	if <*conditional expression*>:
		  <*code block*>
		[elif <conditional expression>:
		  <code block>, …]
		[else:
		  <code block>]
		if x == 1:
		  print(“x is 1”)
		elif x == 2:
		  print(“x is 2”)
		elif x > 3:
		  print(“x is greater than 3”)

		else print("x is not greater than 3, nor is it 1 one or 2")
import	Use code defined in another file without retyping it.	import <*name of module*> import random
in	Used to test whether a given value is one of the elements of an object.	1 in range(10)

is	Used to test whether names reference the same object.	x = None
		x is None # faster than
		x == None

lambda	Shorthand function definition. Usually used where a function needs to be passed as an argument to another function.	lamda <*dummy variables*>: <*expression using dummy variables*> times = lambda x, y: x*y command=lambda x: self.draw_line(self.control_points)
not	Logical negation, used to negate a logical condition. Don't use for testing greater than, less than, or equal.	10 not in range(10)
or	Logical operator to test whether at least one of two things is True.	<*conditional expression*> or <*conditional expression*> x<2 or x>10
pass	Placeholder keyword. Does nothing but stop Python complaining that a code block is empty.	for i in range (10): pass
print	Output text to a terminal.	print("Hello World!")
return	Return from the execution of a function. If a value is specified, return that value, otherwise return None.	return <value or expression> return x+2

while	Execute a code block while the associated condition is True.	while <conditional expression>:
		while True:
		pass

with	Get Python to manage a resource (like a file) for you.	with open(<name of file>,<file mode>) as <object name>:

input – output functions in python

- Real world programs need to be interactive. That means you need to take some sort of input or information from the user and work on that input.
- For example, you would want to take input from the user and then print some results back. We can achieve this using the **input()**function and **print()** function respectively
- We must always remember that the input function takes user's input as a string. So whether your input a number or a string it is treated as a string only.

Example:1

```
print("Enter your name:")

x = input()

print("Hello, " + x)
```

output : Enter your name:

srikanth

Hello, srikanth

Example:2

```
name = input("what's your name :")

age = input("Enter your age:")

print(name + " you are " + age + "  years old")
```

output:

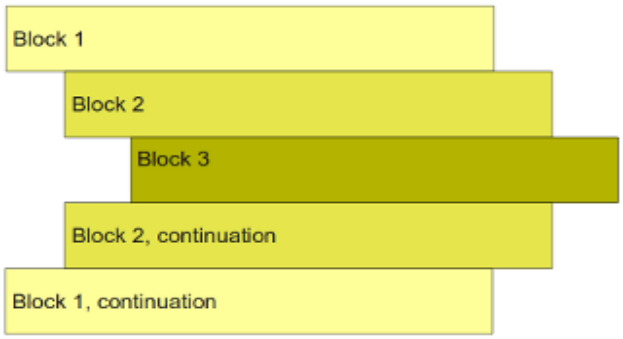
what's your name : srikanth

Enter your age : 28

srikanth you are 28 years old

INDENTATIONS

- White spaces at the beginning of the line is called indentation. The whitespaces or indentations are very important in python
- In most programming languages, indentations has no effect on programming logic. It is used to align statements to make the code readable.
- In python indentation is used to associate and group statements



Example of indentation

Python

```
if       condition          do
if ethernet1 > ethernet2 :
   print “it is bigger!”
indentation
```

The level of indentation groups statements to form a block of statements this means that statements in a block must have the same indentation level. python very strictly checks the indentation level and gives an error if indentation is not correct.

Example 3.9 Program to exhibit indentation errors

```
age = 21
    print("You can vote") # Error! Tab at the start of the line
Traceback (most recent call last):
  File "C:\Python34\Try.py", line 2
    print("You can vote")
    ^
IndentationError: unexpected indent
```

NOTE: the ^ is a symbol that indicates where error has occurred in the program

Indentation

IndentationError: expected an indented block	Compiles

def spam():
pizza = 12
return pizza
print spam()
	def spam():
	  pizza = 12
	  return pizza
	print spam()

All statements in side a block should be at the same indentation level

Example:

```
1
2  for item in range(10):
3      print('I')
4      print('am')
5      print('a')
6      if item % 2 == 0:
7          print('funny')
8          print('and')
9          print('silly')
10     else:
11         print('dull')
12         print('and')
13         print('serious')
14     print('block')
15     print('used')
16     print('as')
17     print('example.')
18
19
20
21
```