Python Programming

Pre-requisites

1	Basic understanding of Computer Programming terminologies.

Agenda

1	Overview of Python and its Features, Installation
2	Basic Syntax and Data Types, Programming constructs
3	Python Dictionary,Tuples,List,Arrays
4	Python Functions and Modules
5	Python File I/O
6	Python Exceptions
7	Python Classes and Objects
8	Python Database Access

Python

Overview of Python and Features

Python

Overview

- **What is Python?**
  - Python is interpreted, interactive, and high level object-oriented high programming language created by Guido Rossum in 1989
  - Python's syntax and *dynamic typing* with its interpreted nature, which makes it an ideal language for scripting and rapid application development.
  - Python supports *multiple programming pattern*, including object oriented, imperative and functional or procedural.
  - Python is not intended to work on special area such as web programming. That is why it is known as *multipurpose* because it can be used with web, enterprise, 3D CAD etc.
  - Python makes the development and debugging fast.

Python Features

Overview

- **Features**
  - Easy to Learn and Use
  - more expressive means that it is more understandable and readable.
  - Python is an interpreted language. Interpreter executes the code line by line at a time. This makes debugging easy and allows interactive testing and debugging of code snippets.
  - Cross-platform Language

    Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.

    Free and Open Source

  Object-Oriented Language

  Extensible

  Large Standard Library

  GUI Programming Support

    Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

  Integrated

  It can be easily integrated with languages like C, C++, JAVA etc.

  Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

Python

Overview

- **Why Python?**
  - Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.).
  - Python has a simple syntax similar to the English language.
  - Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
  - Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
  - Python can be treated in a procedural way, an object-orientated way or a functional way
  - The most common **applications** of **Python** are: web development, scripting, machine learning, and data analysis / data visualization.

Python Applications

Python is known for its general purpose nature that makes it applicable in almost each domain of software development.

1. Web Applications

   We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, beautiful Soup, Feedparser etc. It also provides Frameworks such as Django, Pyramid, Flask etc. to design and develop web based applications. Some important developments are: PythonWikiEngines, Pocoo, PythonBlogSoftware etc.

2. Desktop GUI Applications

   The Kivy is popular for writing multitouch applications.

3. Software Development

   It works as a support language and can be used for build control and management, testing etc.

4. Scientific and Numeric

   Python is popular and widely used in scientific and numeric computing.

5. Business Applications

   Python is used to build Business applications like ERP and e-commerce systems.

Python Applications continued...

5. Console Based Application

   For example: **IPython**
6. Audio or Video based Applications
7. 3D CAD Applications
8. Enterprise Applications

   Python can be used to create applications which can be used within an Enterprise or an Organization. Some real time applications are: OpenErp, Tryton, Picalo etc.
9. Applications for Images

   Using Python several application can be developed for image. Applications developed are: VPython, Gogh, imgSeek etc.

Python

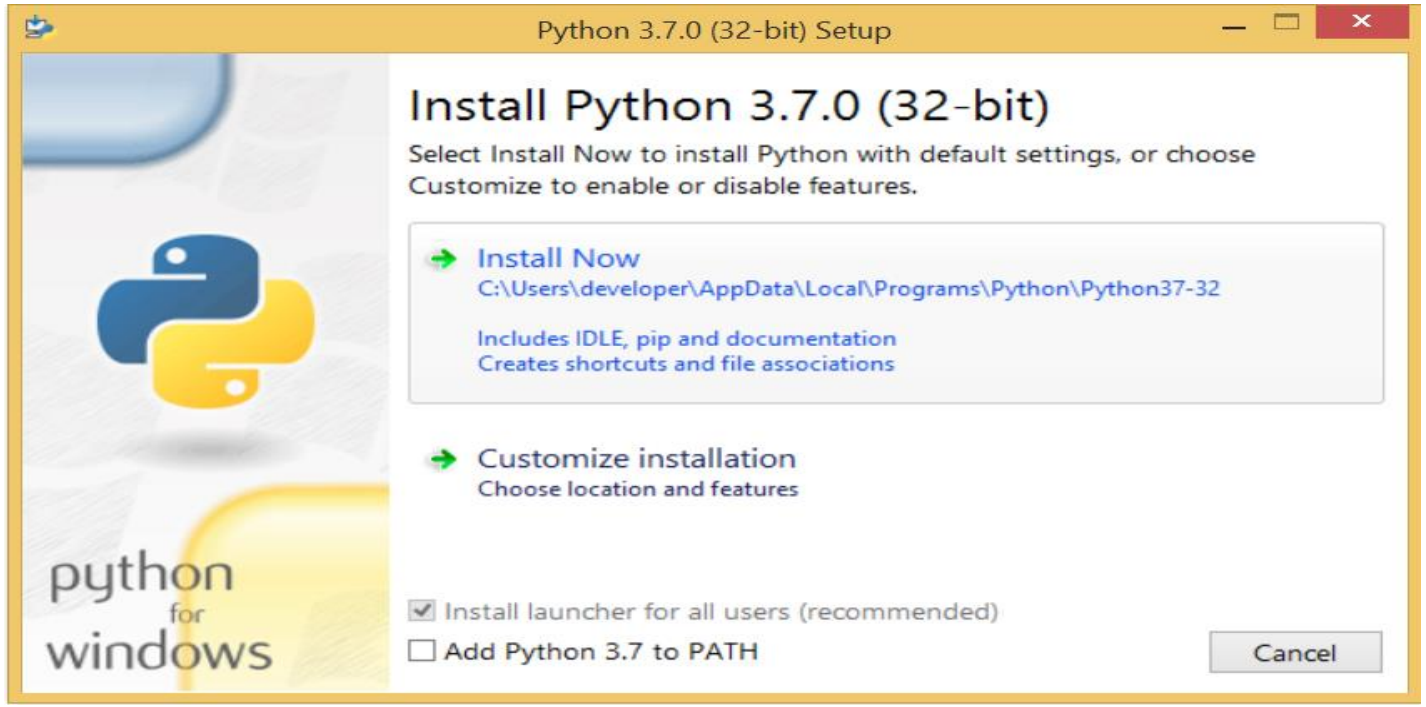
Getting Started

Python Installation

1. Visit the link *https://www.python.org/downloads/* to download the latest release of Python

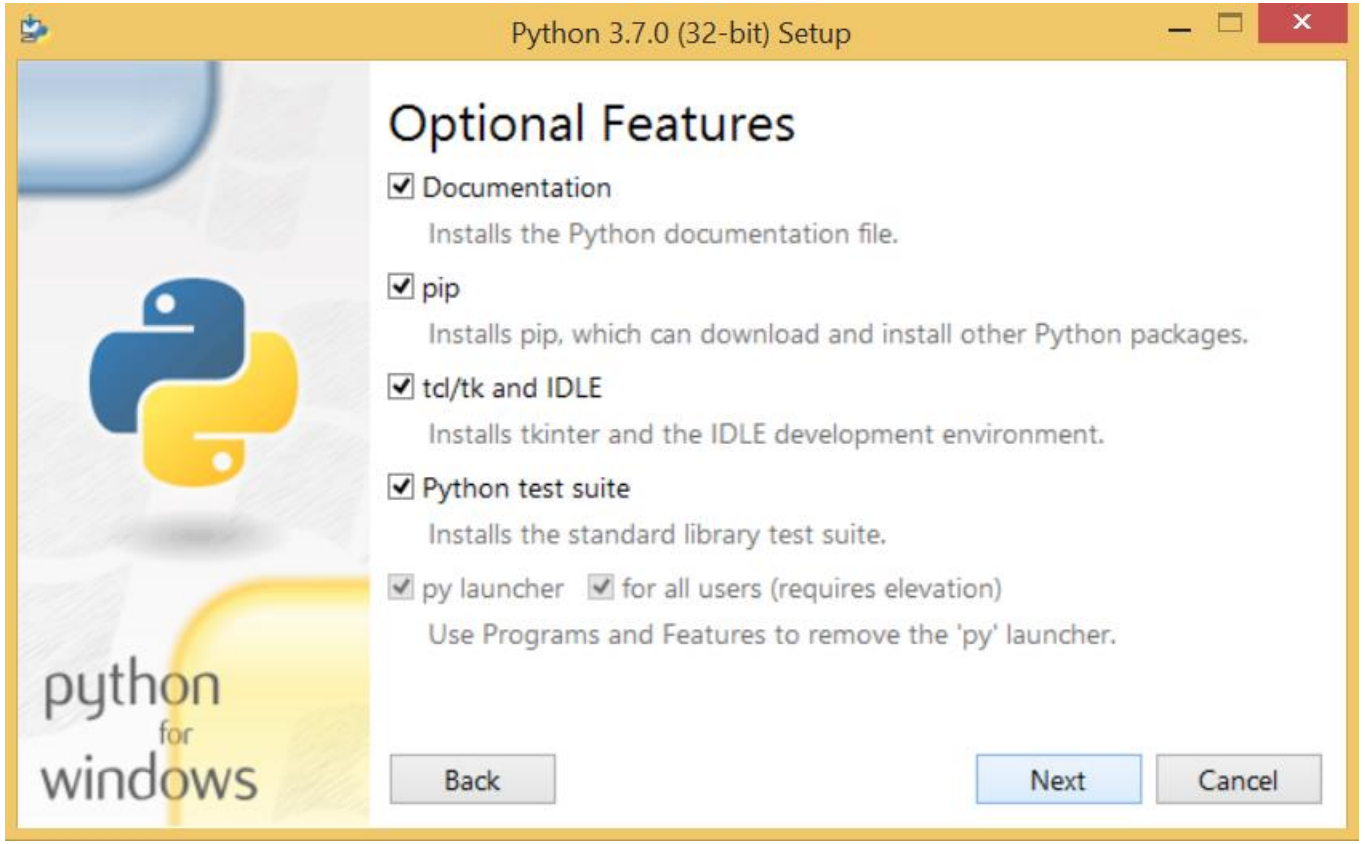
   Here we will install Python 3.6.7 for your respective operating system.

2. Double-click the executable file which is downloaded; the following window will open. Select Customize installation and proceed.



Python Installation

1 Next window shows all the optional features. All the features need to be installed and are checked by default; we need to click next to continue, next window will ask to select the location to install the same and finish the installation.

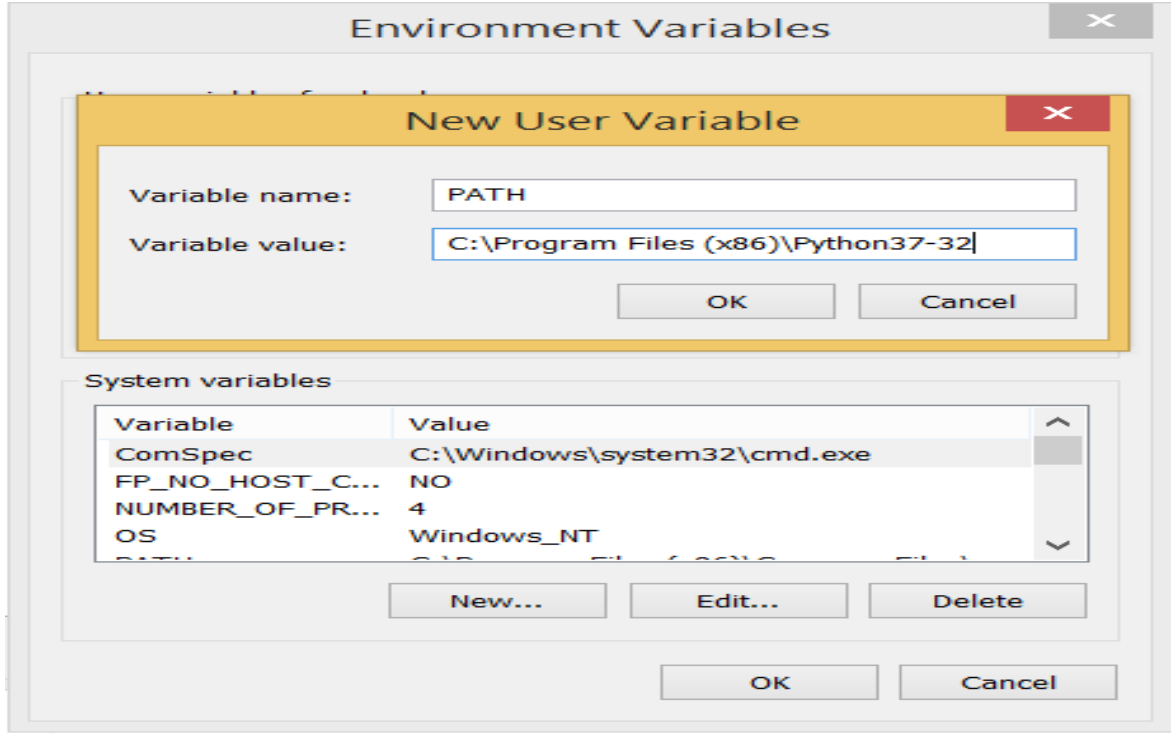


Python Syntax

You can run Python form command prompt.
Python syntax can be executed by writing directly in the Command Line:

Type the command **python** in case of python2 or python3 in case of **python3**. It may show an error , in that case we need to set the path as environment variable.

Restart CMD, and type **python** again. It will open the python interpreter shell where we can execute the python statements.



Python First Program

- Python provides us the two ways to run a program:
  - Using Interactive interpreter prompt
  - Using a script file

- You can run Python from command prompt. Type

Python syntax can be executed by writing directly in the Command Line:

```
>>> print("Hello, World!")
Hello, World!
```

Or by creating a python file on the server, using the .py file extension, and running it in the Command Line:

C:\Users*Your Name*>python myfile.py

You cam mention Python in case of Python 2 and Python3 in case Python 3installed on your system.

Or by using IDLE

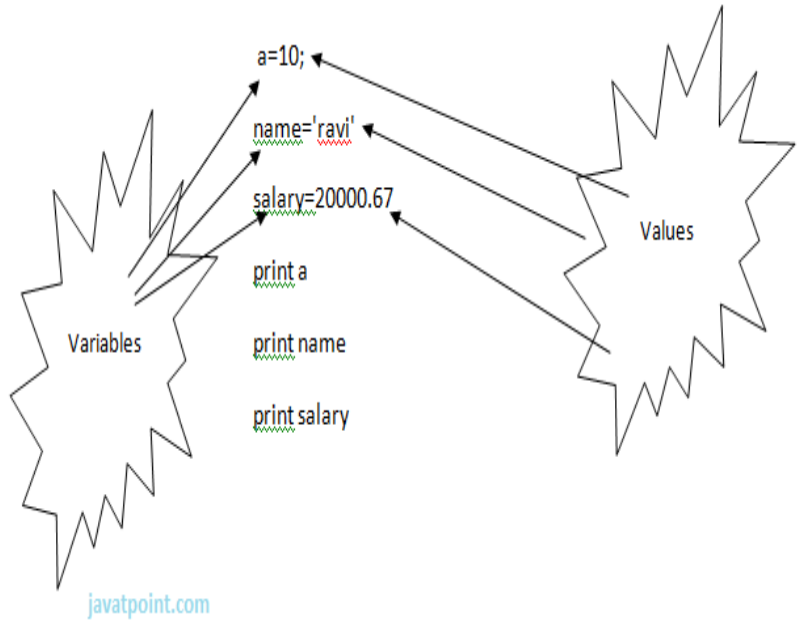
Python Variables

- In Python, we don't need to specify the type of variable because Python is a type infer language and smart enough to get variable type.
- Python is case sensitive.
- It is recommended to use lowercase letters for variable name.
- Python does not bound us to declare variable before using in the application. It allows us to create variable at required time.
- **Multiple Assignment**
  - x=y=z=50
- **Assigning multiple values to multiple variables:**
  - a,b,c=5,10,15
  - **Python Comments**
  - Comments start with a #->Single line comment
    - #This is a comment.

- Python also has extended documentation capability, called docstrings.

  Docstrings can be one line, or multiline.

  Python uses triple quotes at the beginning and end of the docstring:
  - """This is a
    multiline docstring."""



Python Data Types

String

- **Types of Strings:**
- Single line String- Strings that are terminated within a single line are known as Single line Strings.
  - **Example:** text1='hello'
- Multi line String- A piece of text that is spread along multiple lines is known as Multiple line String.
  - There are two ways to create Multiline Strings:
  - Adding black slash at the end of each line.
  - Using triple quotation marks

- **Number**
  - Int x = 1 # int
    - Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.
  - Float y = 2.8 # float
  - Complex z = 1j # complex
    - Complex numbers are written with a "j" as the imaginary part
    - To get access to complex equivalents of the math module and used for advance mathematical feature.

Python Data Types

- **List**
  - Array of elements of similar/different data types and mutable
- **Tuple**
  - Tuple is another form of collection where different type of data can be stored.
  - It is similar to list where data is separated by commas. Only the difference is that list uses square bracket and tuple uses parenthesis.
  - Tuples are enclosed in parenthesis and cannot be changed.
- **Set**
  - Unordered collection of unique value
- **Dictionary**
  - Dictionary is a collection which works on a key-value pair.
  - It works like an associated array where no two keys can be same.
  - Dictionaries are enclosed by curly braces ({}) and values can be retrieved by square bracket([]).

Python Functions and Modules

- **Functions**
  - Function in python can be defined using def statement. It must be followed by the function name and the parenthesized list of formal parameters. The statements that form the body of the function start at the next line, and must be indented.

  - def <<function name>>(parameters):
    "function_docstring "#this is optional
    function suite
    return expression

    For example
    def printMessage(str):
    print(str)
    return
    Calling a function
    printMessage("Hi this is python function")

Python Functions & Modules

- **The Anonymous Functions**
  - These functions are called anonymous because they are not declared in the standard manner by using the *def* keyword and can be defined by lambda functions.

  - Syntax
    - Lambda[arg1,arg2,…]:expression
    - Example:

      Sum=lambda arg1,arg2:arg1+arg2

Object Oriented Concepts

- **Class**
  - A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
- **Class variable**
  - A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods.
- **Function overloading**
  - The assignment of more than one behavior to a particular function.
- **Object**
  - A unique instance of a data structure that's defined by its class. An object comprises both data members (class variables and instance variables) and methods.
- **Operator overloading**
  - The assignment of more than one function to a particular operator.

Object Oriented Concepts

- **Creating Classes**

```
class ClassName:
    Optional doc-strings
    Class suite
```

» **Example:**

```
class Employee:
        empCount=0#class variable
def __init__(self,name,salary):                #constructor
        self.name=name
        self.salary=salary
        Employee. EmpCount+=1
def  displayCount():
        print(Employee.empCount)
```

- **Creating Instance Objects**
  - Emp1=Employee("zara",23445)
- **Set**
  - Unordered collection of unique value
- **Dictionary**
  - Dictionary is a collection which works on a key-value pair.
  - It works like an associated array where no two keys can be same.
  - Dictionaries are enclosed by curly braces ({}) and values can be retrieved by square bracket([]).

Object Oriented Concepts

- **Built-In Class Attributes**

Every Python class keeps following built-in attributes and they can be accessed using dot operator like any other attribute

  - **__dict__** – Dictionary containing the class's namespace.
  - **__doc__** – Class documentation string or none, if undefined.
  - **__name__** – Class name.
  - **__module__** – Module name in which the class is defined. This attribute is "__main__" in interactive mode.
  - **__bases__** – A possibly empty tuple containing the base classes, in the order of their occurrence in the base class list.

- **Destroying Objects (Garbage Collection)**

  - def __del__(self):
    Print class_name,”destroyed”

Class Inheritance

```
class Parent:        # define parent class
   parentAttr = 100
   def __init__(self):
      print "Calling parent constructor"

   def parentMethod(self):
      print 'Calling parent method'

   def setAttr(self, attr):
      Parent.parentAttr = attr

   def getAttr(self):
      print "Parent attribute :", Parent.parentAttr

class Child(Parent): # define child class
   def __init__(self):
      print "Calling child constructor"

   def childMethod(self):
      print 'Calling child method'
```

Object Oriented Concepts

- **Overriding Methods**

```
class Parent:        # define parent class
   def myMethod(self):
      print 'Calling parent method'

class Child(Parent): # define child class
   def myMethod(self):
      print 'Calling child method'

c = Child()          # instance of child
c.myMethod()         # child calls overridden method
```

Object Oriented Concepts

- **Data Hiding**

An object's attributes may or may not be visible outside the class definition. You need to name attributes with a double underscore prefix, and those attributes then are not be directly visible to outsiders.

```
class JustCounter:
   __secretCount = 0

   def count(self):
      self.__secretCount += 1
      print self.__secretCount

counter = JustCounter()
counter.count()
counter.count()
print counter.__secretCount
```

File I/O

- **Reading Keyboard Input**
  - Python provides two built-in functions to read a line of text from standard input, which by default comes from the keyboard. These functions are –
    - raw_input
    - input
  - **The *raw_input* Function**
  - The *raw_input([prompt])* function reads one line from standard input and returns it as a string (removing the trailing newline).
  - str = raw_input("Enter your input: ");
  - print "Received input is : ", str

  - **The *input* Function**
  - The *input([prompt])* function is equivalent to raw_input, except that it assumes the input is a valid Python expression and returns the evaluated result to you.

    str = input("Enter your input: ");

    print "Received input is : ", str

File I/O

- **Opening and Closing Files**
- **The *open* Function**
  - This function creates a **file** object, which would be utilized to call other support methods associated with it.
  - file object = open(file_name [, access_mode][, buffering])

  - **file_name** – The file_name argument is a string value that contains the name of the file that you want to access.
  - **access_mode** – The access_mode determines the mode in which the file has to be opened, i.e., read, write, append, etc. A complete list of possible values is given below in the table. This is optional parameter and the default file access mode is read (r).
  - **buffering** – If the buffering value is set to 0, no buffering takes place. If the buffering value is 1, line buffering is performed while accessing a file. If you specify the buffering value as an integer greater than 1, then buffering action is performed with the indicated buffer size. If negative, the buffer size is the system default(default behavior).

    ```
    fo = open("foo.txt", "wb")
    print "Name of the file: ", fo.name
    print "Closed or not : ", fo.closed
    print "Opening mode : ", fo.mode
    print "Softspace flag : ", fo.softspace
    ```
    fo.close()#The close() method of a *file* object flushes any unwritten information and closes the file object, after which no more writing can be done.

File I/O

- **Reading and Writing Files**
  - fileObject.write(string);#The *write()* method writes any string to an open file.
  - fileObject.read([count]);#The *read()* method reads a string from an open file.
- **File Positions**

File I/O

- **File Positions**

Method	Action
tell()	Gives the current position within the file; in other words, the next read or write will occur at that many bytes from the beginning of the file.
The *seek(offset[, from])*	changes the current file position. The *offset* argument indicates the number of bytes to be moved. The *from* argument specifies the reference position from where the bytes are to be moved.
rename()	The rename() method takes two arguments, the current filename and the new filename.
remove()	You can use the remove() method to delete files by supplying the name of the file to be deleted as the argument.
getcwd()	The getcwd() method displays the current working directory.

Thank You

Atos | Syntel