Applications of python programming

Python is high-level general purpose programming language that is used to develop a wide range of applications including image processing, text processing, web and enterprise level applications using scientific and numeric data from network.

1. System utilities (system Admin tools, Commandline Programs)
2. web Development.
3. Graphical user interfaces (Tkinter, gtk, Qt).
4. Internet scripting
5. Embedded scripting.
6. Database access and programming.
7. Game programming.
8. Rapid prototyping and development.
9. Distributed programming.

Embedded scripting language:- Python is used as an embedded scripting language for various testing / building / deployment / monitoring frameworks, scientific apps, and quick scripts.

3D Software:- 3D software like maya uses python for automating small user tasks, or for doing more complex integration such as talking to databases and asset management systems.

web development:- python is an easily extensible language that provides good integration with database and other web standards. therefore, it is a popular language for web development. For example Quora, odoo and Google App engine are popular web applications based on python.

- GUI-based desktop applications :- Simple syntax, modular architecture, rich text processing tools, and the ability to work on multiple operating systems makes python a preferred choice for developing desktop-based applications. For this, python has various GUI toolkits like wxPython, PyQt or PyGtk which help developers create highly functional Graphical User Interface (GUI) including,
  - → Image processing and graphic design applications
  - → scientific and computational applications.
- Games :- python has various modules, libraries, and platforms that support development of games. while Pysoy is a 3D game engine, PyGame on the other hand provides functionality and a library for game development. Games like Civilization-IV, Disney's Toontown online, Vega strike etc. are coded using python.
- Enterprise and business applications :- Simple and reliable syntax, modules and libraries, extensibility and scalability together make python a suitable coding language for customizing larger applications. For example, Reddit which was originally written in Common Lips, was rewritten in python in 2005. A large part of youtube code is also written in python.
- operating Systems :- python forms an integral part of Linux distributions. For example, Ubuntu's Ubiquity installer, and Fedora's and Red Hat interprisers are written in python.
- Language Development :- python's design and module architecture is used to develop other languages. For example, Boo language uses an OOP model, Syntax and indentation, similar to python.

- prototyping :- Since python is very easy to learn and an open source language, it is widely used for prototype development moreover, agility, extensibility and scalability of code written in python supports faster development from initial prototype.

- Network programming :- python is used for network programming as it has easy to use socket interface, functions for email processing and support for FTP, IMAP, and other Internet protocols.

- Teaching :- Python is a perfect language for teaching programming skills as the introductory as well as advanced level.

ASSIGNMENT

In Python, Programmers need not explicitly declare variables to reserve memory space. The declaration is done automatically when a value is assigned to the variable using the equal sign (=). The operand on the left side of equal sign is the name of the variable and the operand on its right side is the value to be stored in that variable.

In python, we can reassign variables as many times as we want to change the value stored in them. we may even store value of one data type in a statement and then a value of another data in a subsequent statement. This is possible because python variables do not have specific types. So we can assign an integer to a variable, and later assign a string to the same variable. For example,

```
val = 'Hello'
print (val)
val = 100
print (val)
val = 12.34
print (val)
```

OUTPUT:

```
Hello
100
12.34
```

Multiple Assignment :- Python allows Programmers to assign a single value to more than one variable simultaneously. For example:

```
sum = flag = a = b = 0.
```

In the above statement, all four integer variables are assigned a value 0. we can also assign different values to multiple variables simultaneously as shown below:

```
sum, a, b, mesg = 0, 3, 5, "RESULT"
```

Here, variable sum, a, b are integers and mesg is a string. sum is assigned a value 0, a is assigned 3, b is assigned 5 and mesg is assigned "RESULT".

Remember that trying to reference a variable that has not been assigned any value causes an error.

Features of python

python is an exciting and powerful language with the right combination of performance and features that makes programming fun and easy. It is a high-level, interpreted, interactive, object-oriented and a reliable language that is very simple and uses English-like words. It has a vast library of modules to support integration of complex solutions from pre-built components.

python is an open-source project, supported by many individuals. It is a platform-independent, scripted language, with complete access to operating system APIs. this allows users to integrate applications seamlessly to create high-powered, highly-focused applications. python is a complete programming language with the following features.

Simple python is a simple and a small language. Reading a program written in python feels almost like reading English this is in fact the greatest strength of python which allows programmers to concentrate on the solution to the problem rather than the language itself.

Easy to Learn A python program is clearly defined and easily readable. the structure of the program is very simple It uses few keywords and a clearly defined syntax. this makes it easy for just anyone to pick up the language quickly.

Versatile Python supports development of a wide range of applications ranging from simple text processing to WWW browsers to games.

Free and open Source python is an example of an open source software. therefore, anyone can freely distribute it, read the source code, edit it, and even use the code to write new programs.

High-level Language when writing programs in Python, the programmers does not have to worry about the low-level details like managing memory used by the program etc. They just need to concentrate on writing solutions of the current problem at hand.

Interactive Programs in Python work in interactive mode which allows interactive testing and debugging of pieces of code. Programmers can easily interact with the interpreter directly at the python prompt to write their programs.

Portable python is a portable language and hence the programs behave the same on a wide variety of hardware platforms and has the same interface on all platforms.

object oriented python supports object-oriented as well as procedure-oriented style of programming. while object-oriented technique encapsulates data and functionalities within objects, procedure-oriented technique, on the other hand, builds the around procedures or functions which are nothing but reusable pieces of programs. python is powerful yet a simple language for implementing OOP concepts.

Interpreted python is interpreted. Interpreted language has a simpler execute cycle and also works faster. python is processed at run-time by the interpreter. So there is no need to compile a program before executing it. Basically, python converts the source code into an intermediate form called bytecode.

Dynamic python executes dynamically. Programs written in python can be copied and used for flexible development of applications. If there is any error, it is reported at run-time to allow interactive program development.

Extensible Since python is an open source software, anyone can add low-level modules to the python interpreter. These modules enables programmers to add to or customize their tools to work more efficiently. Morever, if you want a piece of code not to be accessible for everyone, then you can even code that part of your program in C or C++ and then use them from your python program.

Embeddable programmers can embed python within their C, C++, COM, ActiveX, CORBA and JAVA programs to give 'scripting' capabilities for users.

Extensive Libraries python has a huge library that is easily portable across different platforms. These library functions are compatible on UNIX, Windows, Macintosh, etc. and allows programmers to perform a wide range of applications varying from text processing, maintaining databases, to GUI programming.

Easy Maintenance code written in python is easy to maintain.

Secure The python language environment is secure from tampering. Modules can be distributed to prevent altering the source code. Apart from this, additional security checks can be easily added to implement additional security features.

Robust Python programmers cannot manipulate memory directly. Moreover, errors are raised as exceptions that can be catch and handled by the program code. for every syntactical mistake, a simple and easy to interpret message is displayed. All these things makes the language robust.

Multi-threaded python supports multi-threading, that is executing more than one process of a program simultaneously. It also allows programmers to perform process management tasks.

Garbage collection the python run-time environment handles garbage collection of all python objects. For this, a reference counter is maintained to assure that no object that is currently in use is deleted. An object that is no longer used or has gone out of slope are eligible for garbage collection. This frees the programmers from the worry of memory leak and dangling reference problems. However, the programmer can still perform memory management functions by explicitly deleting an unused objects.

IDLE User Interface

IDLE provides developing environment for python programs.

IDLE means Integrated Development and Learning environment.

IDLE has the following features:

- coded in 100% pure python, using the tkinter GUI Toolkit
- cross-platform: works mostly the same on windows, UNIX, and Mac OS X
- python shell window (interactive interpreter) with colorizing of code input, output and error messages.
- Multi-window text editor with multiple undo, python colorizing, smart indent, call tips, auto completion and other features.
- Search within any window, replace within editor windows and search through multiple files (grep).
- debugger with presistent breakpoints, stepping and viewing of global and local namespaces.
- Configuration, browsers, and other dialogs.
- Auto-indent and unindent for python code in the editor.
- word auto-completion while typing, invoked by a Tab press.
- Ballon help pop ups for a function call when you type its opening
- Pop-up selection lists of object attributes when you type a . after an object's name.
- and either pause or press Tab.

Apart from IDLE, here some of python's most commonly used IDEs :-

1. Eclipse and PyDev
2. Komodo
3. NetBeans IDE for python
4. Pythonwin
5. Wing, Visual Studio and others.

input-output in python

Real world programs need to be interactive. With interactive we mean that we need to take some sort of input or information from the user and work on that input.

Python 2	Python 3
① raw_input()	① input()
② input()	

INPUT OPERATION

To take input from the users, python makes use of the input() function. The input() function prompts the user to provide some information on which the program can work and give the result. However, we must always remember that the input function takes user's input as a string. So whether you input a number or a string, it is treated as a string only. The syntax of input() method is:

```
input([prompt])
```

The input method takes a single optional argument:

→ prompt (optional) a string that is written to standard output (usually screen) without trailing newline.

→ Return value from input() The input() method reads a line from input, converts the line into a string by removing the trailing newline, and returns it. If EOF is read, it reasies an EOFError Exception.

Example 1:

```
# get input from the User
name = input("Enter name")
print('the input string is:', name).
```

Example 2:

```
x = int(input("Enter First number:"))
Y = int(input("Enter second number:"))
print("the sum:", x+y)
```

(or)

```
print("the sum:", int(input("Enter First number:"))+int(
    input("Enter second number:")))
```

Example 3:

```
eno = int(input("Enter Employee Number:"))
ename = input("Enter Employee Name:")
esal = float(input("Enter Employee Salary:"))
eaddr = input("Enter Employee Address:")
married = bool(input("Enter Employee Marriage status:
                     [True/False]"))
print("please Confirm Information...")
print("Employee Number:", eno)
print("Employee Name:", ename)
print("Employee Salary:", esal)
print("Employee Address:", eaddr)
print("Employee Marriage?:", married).
```

Python Programming

UNIT-1 : Introduction : History of Python, Need of python programming, Applications of python programming using the REPL(Shell), Running Python scripts, variables, Assignment, Keywords, input-output, Indentation.

Introduction to python

python is a general purpose, interpreted, object-oriented Programming language.

- python is a general purpose programming language that means we can use python to write code for any programming task.
- python is interpreted, it means that python code is translated and executed by an interpreter, one statement at a time.
- python is object-oriented programming language (oop). Data in python are objects created from classes. A class is essentially a type or category that defines objects of the same kind with properities and methods for manipulating objects. oop is powerful tool for developing reusable software.

Python is now used in Google search engine, in mission-critical projects at NASA, and in transaction processing at the Newyork stock Exchange.

Python has become popular programming Language widely used in industry and academia due to its simple, concise (brief), and sensitive syntax and extensive library.

- python is great for backend web development, data analysis, aritificial intelligence and scientific computing.
- python supports multiple programming paradigms, including object-oriented, imperative, functional programming and procedural styles.
- python interpreters are available for many operating systems, allowing python code to run on a wide variety of systems and python is open source software.
- python uses dynamic typing and a mix of reference counting and a cycle-detecting garbage collector for memory management. An important feature of python is dynamic name resolution (late binding), which binds method and variable names during program execution.

Keywords

Keywords are predefined, reserved words that are used in programming and they have a special meaning. These reserved words cannot use them as constant or variable or any other identifier names. python has thirty three (33) Keywords :-

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

In python, Keywords are case sensitive.

To print the Keywords in python, the following code has to be executed:

```
>>> import keyword
>>> print(keyword.kwlist)
```

Limitations of Python

→ parallel processing can be done in python but not as elegantly as done in some other languages.

→ Being an interpreted language, python is slow as compared to c/c++. python is not a very good choice for those developing a high-graphic 3d game that takes up a lot of cpu.

→ As compared to other languages, python is ~~slow as~~ evolving continuously and there is little substantial documentation available for the language.

→ As of now, there are few users of python as compared to those using c, c++ or Java.

→ It lacks true multiprocessor support.

→ It has very limited commercial support point.

→ python is slower than C or c++ when it comes to computation of heavy tasks and desktop applications.

→ It is difficult to pack up a big python application into a single executable file. This makes it difficult to distribute python to non-technical users.

LITERALS IN PYTHON

Literals can be defined as a data that is given in a variable or constant. Python supports the following literals:

① String literals:

String literals can be formed by enclosing a text in the quotes. We can use both single or double quotes for a string. examples are:

"Viit", '12345'

② Numeric literals:

Numeric literals are immutable. Numeric literals belong to following four different numerical types:

① int ② float ③ Complex ④ long

examples are:

100, 87032845L, -26.2, 3.14j

③ Boolean literals:

A boolean literals can have any of the two values; True or False.

④ Special literals:

Python contains one special literal i.e None. It is used to specify to that field that is not created.

⑤ Literal collections:

collections such as tuples, lists and Dictionary are used in python.

Need of python programming

1. Python is Great for Beginners

Beginners learn python very quickly.

2. Web Development with python.

- What once took me hours in PHP could be done in minutes using python. Not to mention the fact that my code was a lot faster than more stable.

3. Iterative, responsive Design

- start with an idea, then refine the idea and product until you have "made it". python is ideal for this process this language allows you to code quickly, building complex applications with minimal lines of code.

4. python has High Salaries.

python is second highest salary average developer in 2016 at USA (First is Swift)

5. python, Artificial Intelligence & Machine Learning

python is the future of Aritificial Intelligence and Machine Learning. python released the numerical calculation engines such as Numpy and SciPy, allowing complex calculations to be done by a single "import" statement followed by a function call.

6. Great Community and Support.

Python has a large supporting community. There are numerous active forums online which can be handy if you stuck. (Learn python subreddit, Google Forum for python, etc.)

print() in python

The print() function prints the given object to the standard output device (screen) or to the text stream file.

Syntax :-

```
print(*objects, sep=' ', end='\n', file=sys.stdout,
      flush=False)
```

Parameters :-

→ objects – object to be printed. where * indicates that there may be more than one object.

→ sep – objects are separated by sep. Default value : ' '

→ end – end is printed at last. Default value : '\n'

→ file – must be an object with write(string) method. If omitted it, sys.stdout will be used which prints objects on the screen. Default value: stdout.

→ flush – if True, the stream is forcibly flushed. Default value : False

Return value from print()

It doesnot return any value ; it returns None.

Examples :-

①
```
>>> print("a", "b")
a b
```

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

③
```
>>> a = 3.564
>>> print("a = \n", a)
a =
3.564.
```

④
```
>>> print("a", "b", sep="")
ab
```

(5) The output of the print function is send to the standard output stream (sys.stdout) by default. By redefining the keyword parameter "file" we can send the output into a different stream. e.g- sys.stderr or a file.

```
>>> fh = open("data.txt", "w")
>>> print ("42 is the answer, but what is the Question?",
                                    file = fh)
>>> fh.close()
```

(6) We can also possible to redirect the output to the standard error channel this way :-

```
>>> import sys
>>> #output into sys.stderr;
...
>>> print ("error: 42", file = sys.stderr)
```

Python provides us with an alternative method to have more control over the look of our output is called ~~feral~~ formatted strings. A formatted string is a template in which words or spaces that will remain constant are combined with placeholders for variables that will be inserted into the string.

ex:- `print("%s is %d years old." % (Name, age))`

This simple example illustrates a new string expression. The % operator is a string operator called format operator. The left side of the expression holds the template or format string and the right side holds a collection of values that will be substituted into the format string.

Python Programming using the REPL (Shell)

A Read - Eval - Print - Loop (REPL), also known as an interactive toplevel or language shell, is a simple, interactive computer programming environment that takes single user inputs (i.e. single expressions), evaluates them, and returns the result to the user; a program written in REPL environment is executed piecewise.

REPL: this is a procedure that just loops, accepts one command at a time, executing it, and printing the result. The three steps at each iteration of the loop are:

1. Calling read to read the characters that make up a textual expression from the keyboard input buffer, and construct a data structure to represent it.
2. Calling eval to evaluate the expression - intuitively, eval "figures out what the expression means" and "does what it says to do", returning the value of the expression - and
3. Calling write to print a textual representation of the resulting from eval, so that the user can see it.

You can write your own read-eval-print loop for your own programs, so that users can type in expressions, and you can interpret them any way you want. We can start up read-eval-print loop (by typing in (rep-loop)), and it will take over from the normal scheme read-eval-print loop - interpreting expressions your way.

Here a very simple read-eval-print loop:

```
(define (rep-loop)
 ( display "repl>" ) ; print a prompt
 ( write (eval (read))) ; read expr., pass to eval, write result
 (rep-loop))          ; loop (tail-recursive call) to do it again.
```

Interactive shell :- An interactive shall reads Commands from user input on a terminal. Among other things, such as a shell reads startup files on activation, displays a prompt, and enables job control by default. The user can interact with the shell. That's how the interactive shell name came into being.

Characterstics of REPL :-

→ Auto-indent : when typing python statements indentation will be done automatically.

→ Auto-Completion: while typing a command at the REPL, if the line typed so far corresponds to the beginning of the name of something, then pressing TAB will show possible things that could be entered.

→ Interrupting a running program: we can interrupt a running program by pressing Ctrl-C.

→ Paste mode : If we want to paste some code into your terminal window, the auto-indent feature will mess things up

→ Soft Reset : It will reset the python interpreter.

Running python scripts

when we instruct to run our script, there are a few steps that python carries out before our code actually starts crunching away. Specifically, it's first compiled to something called byte code and then routed to something called virtual machine.

python first compiles our source code (.py) into a format known as bytecode. Compilation is simply a translation step, and byte code is a lower-level, platform independent representation of our source code. Roughly, python translates each of our source statements into a group of byte code instructions by decomposing them into individual steps. This byte code translation is performed to speed execution - byte code can be run much more quickly than the original source code statements in our text file.

one our program has been compiled to byte code (.pyc files), it is shipped off for execution to something generally known as the python virtual machine (PVM)

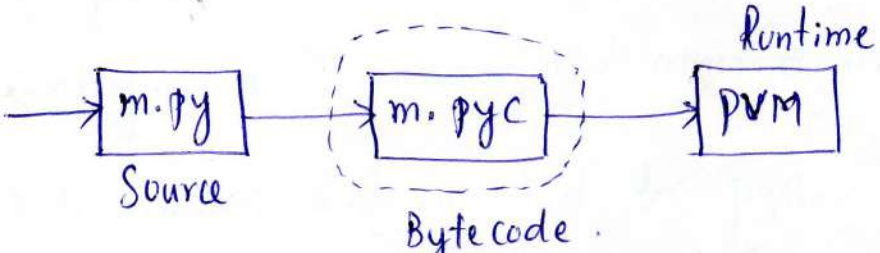

fig. python's traditional runtime execution model.

Source code you type is translated to byte code, which is then run by the python virtual machine. Our code is automatically compiled, but then it is interpreted.

Perhaps the simplest way to run python programs is to type them at python's interactive command line, sometimes called the interactive prompt. There are a variety of ways to start this command line: in an IDE, from a system console, and so on.

VARIABLES AND IDENTIFIERS

Variables play a very important role in most programming languages, and python is no exception. Variables are nothing but reserved memory locations to store values. Variables means its value can vary. we can store any piece of information in a variable. A variable can be identified easily by its name. Every variable is assigned a name which can be used to refer to the value later in the program. The syntax is as follows:

```
variable = expression
```

Variables are examples of identifiers. Identifiers as the name suggests, are names given to identify something. this something can be a variable, function, class, module, or other object. For naming any identifier, there are some basic rules that we must follow. these rules are:

→ The first character of an identifier must be an underscore '_' or a letter (upper or lowercase)

→ The rest of the identifier name can be underscores '_', letters (upper or lowercase), or digits (0-9).

→ Identifier names are case-sensitive. For example, myvar and myVar are not the same.

→ Punctuation characters such as @, $ and % are not allowed within identifiers.

→ Keywords cannot be used as identifiers.

Examples of valid identifier names are sum, _my_Var, num1, x, var_20 etc

Examples of invalid identifier names are 1num, my-var, %check, Basic sal, H#R&A etc.

→ Blank spaces or while spaces are not allowed.