****

**About Python**

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

## Features of Python

### Simple

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.

### Easy to Learn

As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

### Free and Open Source

Python is an example of a *FLOSS* (Free Libre and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

### 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, etc.

### Portable

Due to its open-source nature, Python has been ported to (i.e. changed to make it work on) many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features.

You can use Python on GNU/Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE and PocketPC!

You can even use a platform like [Kivy](http://kivy.org/" \t "_blank) to create games for your computer *and* for iPhone, iPad, and Android.

### Interpreter

A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a language that is spoken by your computer (binary code i.e. 0’s and 1’s) using a compiler with various flags and options. When you run the program, the linker/loader software copies the program from hard disk to memory and starts running it.

Python, on the other hand, does not need compilation to binary. You just *run* the program directly from the source code. Internally, Python converts the source code into an intermediate form called byte codes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded etc. This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

### Object Oriented

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. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

### Extensible

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 it from your Python program.

### Embeddable

You can embed Python within your C/C++ programs to give *scripting* capabilities for your program's users.

### Extensive Libraries

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, FTP, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), and other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the *Batteries Included* philosophy of Python.

Besides the standard library, there are various other high-quality libraries which you can find at the [Python Package Index](http://pypi.python.org/pypi).

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Programs** | **Page no:** |
| 1 | Write a program to print “Hello World!” to the output console. |  |
| 2 | Write a program to read something from the input console and write it to the output console. |  |
| 3 | Write a program to find the sum and difference of two numbers. |  |
| 4 | Write a program to swap the contents of two variables with a temporary variable. |  |
| 5 | Write a program to swap the contents of two variables without a temporary variable. |  |
| 6 | Write a program to find the largest number among two numbers. |  |
| 7 | Write a program to find the largest number among three numbers. |  |
| 8 | Write a program to find the largest string (size wise & alphabetical wise) among two strings. |  |
| 9 | Write a program to input marks of a student and to find the percentage & grade as per the SMVDU norms. |  |
| 10 | Write a program to print the multiplication table of a number up to a range. |  |
| 11 | Write a program to find the area and perimeter of shapes (triangle, rectangle and circle). |  |
| 12 | Write a program to calculate the net salary of an employee.  (Net salary = BP+TA+DA+HRA, TA = 5% BP, DA = 10% BP, HRA = 15% BP) |  |
| 13 | Write a program to find the factors of a number. |  |
| 14 | Write a program to print each digits of a number. |  |
| 15 | Write a program to check whether the given number is odd or even. |  |
| 16 | Write a program to check whether a given number is prime or not. |  |
| 17 | Write a program to check whether a given number is palindrome or not. |  |
| 18 | Write a program to check whether a given number is armstrong or not. |  |
| 19 | Write a program to check whether a given year is leap year or not. |  |
| 20 | Write a program to find the factorial of a number. |  |
| 21 | Write a program to generate the Fibonacci series up to a range. |  |
| 22 | Write a program to generate the first “n” Fibonacci numbers. |  |
| 23 | Write a program to print following pattern.  \*  \*\*  \*\*\*  ……….. |  |
| 24 | Write a program to print the following pattern.  ………..  \*\*\*  \*\*  \* |  |
| 25 | Write a program to print the following pattern.  \*  \*\*  \*\*\*  \*\*  \* |  |
| 26 | Write a program to print the diamond pattern.  \*  \* \*  \* \* \*  \* \*  \* |  |
| 27 | Write a program to print following pattern.  1  12  123  ……….. |  |
| 28 | Write a program to print the following pattern.  ……….  321  32  3 |  |
| 29 | Write a program to print the following pattern.  1  2 3  4 5 6  ……….. |  |
| 30 | Write a program to print the following pattern.  1  2 3  4 5 6  7 8  ……….. |  |
| 31 | Write a program to find the sum of first “n” natural numbers using recursion. |  |
| 32 | Write a program to find the factorial of a number using recursion. |  |
| 33 | Write a program to generate the fibonacci series up to a range using recursion. |  |
| 34 | Write a program to find out the doublet and triplet of a number using lambda functions. |  |
| 35 | Write a program to implement the use of arguments in to a function and return values from a function. |  |
| 36 | Write a program to demonstrate the use of local and global variables. |  |
| 37 | Write a program to implement the usage of some built-in library modules (math, matplotlib, turtle, numpy, random etc.). |  |
| 38 | Write a program to implement the use of user defined library modules. |  |
| 39 | Write a program to print the sine and cosine values for the degrees 0, 30, 45, 60 and 90. |  |
| 40 | Write a program to make a list in Python and perform following operations on List:   1. length using len() function 2. print element at index 0 3. adding an elements of a list to another list using + operator 4. appending an element to the list 5. negative indexing in list 6. remove the first occurrence of element a from list 7. reverse the list 8. sort list |  |
| 41 | Write a program to demonstrate list creation, copy the entire list to another list. |  |
| 42 | Write a program to demonstrate slicing operations on the list. |  |
| 43 | Write a program to perform spilt and join operation on a list. |  |
| 44 | Write a program to search an element in a list, find the number of occurrences and the index of its first occurrence. |  |
| 45 | Write a program to create a 2D list and a 3D list. |  |
| 46 | Write a program to iterate over a 2D list in different ways. |  |
| 47 | Write a program to demonstrate the use of numpy library for creating arrays in Python. |  |
| 48 | Write a program to find the sum and difference of two matrices. |  |
| 49 | Write a program to find the product of two matrices. |  |
| 50 | Write a program to demonstrate use of tuple in Python with their inbuilt functions. |  |
| 51 | Write a program to demonstrate use of set in Python with their inbuilt functions. |  |
| 52 | Write a program to demonstrate use of dictionary in Python with their inbuilt functions. |  |
| 53 | Write a program to implement the usage of iterators on collections. |  |
| 54 | Write a program to implement the usage of iterators on strings. |  |
| 55 | Write a program to implement the usage of look up table as a dictionary for finding out the country name from code of 25 countries. |  |
| 56 | Write a program to create a look up table using dictionary for finding the factorial of numbers up to 25. |  |
| 57 | Write a program to implement the concept of class and object creation. |  |
| 58 | Write a program to store marks of “n” students in a class for “m” subjects using OOP. |  |
| 59 | Write a program to create a class for animals and to create sub-classes for birds, dogs and humans. |  |
| 60 | Write a program to create a class for polygons and to create sub-classes for triangles, rectangles, squares, pentagons etc. |  |
| 61 | Write a program to create a class for trees and to create sub-classes like fruits, dry fruits, juices etc. (multi-level inheritance) |  |
| 62 | Write a program to create a class for students and this class should inherit the properties of all departments. (multiple inheritance) |  |
| 63 | Write a program to create all shapes in a canvas using tkinter library. |  |
| 64 | Write a program to create a table of 3\*3 square cells in a canvas and to colour each cell differently using tkinter library. |  |
| 65 | Write a program to prepare a simple registration form using tkinter library. |  |

**Signature of Faculty in charge**