Computer Science Class -XI

Code No. 083

2023-24

1. Learning Outcomes

Students should be able to:

a) develop basic computational thinking
b) explain and use data types
c) appreciate the notion of algorithms
d) develop a basic understanding of computer systems- architecture, operating system, and cloud computing
e) explain cyber ethics, cyber safety, and cybercrime
f) understand the value of technology in societies along with consideration of gender and disability issues.

2. Distribution of Marks

Unit No.	Unit Name	Marks	Periods	
			Theory	Practical
I	Computer Systems and Organisation	10	10	10
II	Computational Thinking and Programming -1	45	80	60
III	Society, Law, and Ethics	15	20	—
	Total	**70**	**110**	**70**

3. Unit wise Syllabus

Unit I: Computer Systems and Organisation

- Basic computer organisation: Introduction to Computer System, hardware, software, input device, output device, CPU, memory (primary, cache and secondary), units of memory (bit, byte, KB, MB, GB, TB, PB)
- Types of software: System software (Operating systems, system utilities, device drivers), programming tools and language translators (assembler, compiler, and interpreter), application software
- Operating System(OS): functions of the operating system, OS user interface
- Boolean logic: NOT, AND, OR, NAND, NOR, XOR, NOT, truth tables and De Morgan's laws, Logic circuits
- Number System: Binary, Octal, Decimal and Hexadecimal number system; conversion

between number systems

- Encoding Schemes: ASCII, ISCII, and Unicode (UTF8, UTF32)

Unit II: Computational Thinking and Programming - I

- Introduction to Problem-solving: Steps for Problem-solving (Analyzing the problem, developing an algorithm, coding, testing, and debugging), representation of algorithms using flowchart and pseudocode, decomposition
- Familiarization with the basics of Python programming: Introduction to Python, Features of Python, executing a simple “hello world" program, execution modes: interactive mode and script mode, Python character set, Python tokens(keyword, identifier, literal, operator, punctuator), variables, concept of l-value and r-value, use of comments
- Knowledge of data types: Number(integer, floating point,complex), boolean, sequence(string, list, tuple), None, Mapping(dictionary), mutable and immutable data types.
- Operators: arithmetic operators, relational operators, logical operators, assignment operators, augmented assignment operators, identity operators (is, is not), membership operators (in not in)
- Expressions, statement, type conversion, and input/output: precedence of operators, expression, evaluation of an expression, type-conversion (explicit and implicit conversion), accepting data as input from the console and displaying output.
- Errors- syntax errors, logical errors, and run-time errors
- Flow of Control: introduction, use of indentation, sequential flow, conditional and iterative flow
- Conditional statements: if, if-else, if-elif-else, flowcharts, simple programs: e.g.: absolute value, sort 3 numbers and divisibility of a number.
- Iterative Statement: for loop, range(), while loop, flowcharts, break and continue statements, nested loops, suggested programs: generating pattern, summation of series, finding the factorial of a positive number, etc.
- Strings: introduction, string operations (concatenation, repetition, membership and slicing), traversing a string using loops, built-in functions/methods–len(), capitalize(), title(), lower(), upper(), count(), find(), index(), endswith(), startswith(), isalnum(), isalpha(), isdigit(), islower(), isupper(), isspace(),lstrip(), rstrip(), strip(), replace(), join(), partition(), split()
- Lists: introduction, indexing, list operations (concatenation, repetition, membership and slicing), traversing a list using loops, built-in functions/methods–len(), list(), append(), extend(), insert(), count(), index(), remove(), pop(), reverse(), sort(), sorted(), min(), max(), sum(); nested lists, suggested programs: finding the maximum, minimum, mean of numeric values stored in a list; linear search on list of numbers and counting the frequency of elements in a list.
- Tuples: introduction, indexing, tuple operations (concatenation, repetition, membership and slicing); built-in functions/methods – len(), tuple(), count(), index(), sorted(), min(), max(), sum(); tuple assignment, nested tuple; suggested programs: finding the minimum, maximum, mean of values stored in a tuple; linear search on a tuple of numbers, counting the frequency of elements in a tuple.
- Dictionary: introduction, accessing items in a dictionary using keys, mutability of a dictionary (adding a new term, modifying an existing item), traversing a dictionary, built-in

functions/methods – len(), dict(), keys(), values(), items(), get(), update(), del(), del, clear(), fromkeys(), copy(), pop(), popitem(), setdefault(), max(), min(), sorted(); Suggested programs: count the number of times a character appears in a given string using a dictionary, create a dictionary with names of employees, their salary and access them.

- Introduction to Python modules: Importing module using 'import <module>' and using from statement, importing math module (pi, e, sqrt(), ceil(), floor(), pow(), fabs(), sin(), cos(), tan()); random module (random(), randint(), randrange()), statistics module (mean(), median(), mode()).

Unit III: Society, Law and Ethics

- Digital Footprints
- Digital Society and Netizen: net etiquettes, communication etiquettes, social media étiquettes
- Data Protection: Intellectual property rights (copyright, patent , trademark), violation of IPR(plagiarism, copyright infringement, trademark infringement), open source software and licensing (Creative Commons, GPL and Apache)
- Cyber Crime: definition, hacking, eavesdropping, phishing and fraud emails, ransomware, cyber trolls, cyber bullying
- Cyber safety: safely browsing the web, identity protection, confidentiality
- Malware: viruses, trojans, adware
- E-waste management: proper disposal of used electronic gadgets.
- Information Technology Act (IT Act)
- Technology and society: Gender and disability issues while teaching and using computers

4. Practical

S.No.	Unit Name	Marks (Total=30)
1.	**Lab Test (12 marks)**	
	Python program (60% logic + 20% documentation + 20%code quality)	**12**
2.	**Report File + Viva (10 marks)**	
	Report file: Minimum 20 Python programs	**7**
	Viva voce	**3**
3.	Project (that uses most of the concepts that have beenlearnt)	**8**

5. Suggested Practical List
Python Programming

- Input a welcome message and display it.
- Input two numbers and display the larger / smaller number.

- Input three numbers and display the largest / smallest number.
- Generate the following patterns using nested loops:

Pattern-1	Pattern-2	Pattern-3
* ** *** **** *****	12345 1234 123 12 1	A AB ABC ABCD ABCDE

- Write a program to input the value of x and n and print the sum of the following series:
  - $1 + x + x^2 + x^3 + x^4 + \cdots x^n$
  - $1 - x + x^2 - x^3 + x^4 - \cdots x^n$
  - $x + \frac{x^2}{2} + \frac{x^3}{3} + \frac{x^4}{4} + \cdots \frac{x^n}{n}$
  - $x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots \frac{x^n}{n!}$
- Determine whether a number is a perfect number, an Armstrong number or a palindrome.
- Input a number and check if the number is a prime or composite number.
- Display the terms of a Fibonacci series.
- Compute the greatest common divisor and least common multiple of two integers.
- Count and display the number of vowels, consonants, uppercase, lowercase characters instring.
- Input a string and determine whether it is a palindrome or not; convert the case of charactersin a string.
- Find the largest/smallest number in a list/tuple
- Input a list of numbers and swap elements at the even location with the elements at the oddlocation.
- Input a list/tuple of elements, search for a given element in the list/tuple.
- Create a dictionary with the roll number, name and marks of n students in a class and displaythe names of students who have marks above 75.

6. Suggested Reading Material

- NCERT Textbook for Computer Science (Class XI)
- Support Material on CBSE website

UNIT -1

INTRODUCTION TO COMPUTER SYSTEM

A computer is an electronic device, under the control of instructions stored in its memory that can accept data (input), process the data according to specified rules(Program) on processor & produces information (output), and store the information for future use.

Data vs Information

Data are raw numbers or other findings which, by themselves, are of limited value.Information is data that has been converted into a meaningfuland useful context.Computers are being used extensively nowadays in
everyday life/every field

In the form of laptop, desktop, smartphone,gadgets etc.

Advantages of computer

- Speed
- Accuracy
- Huge storage
- Versatility
- Tirelessness

Disadvantages of computer

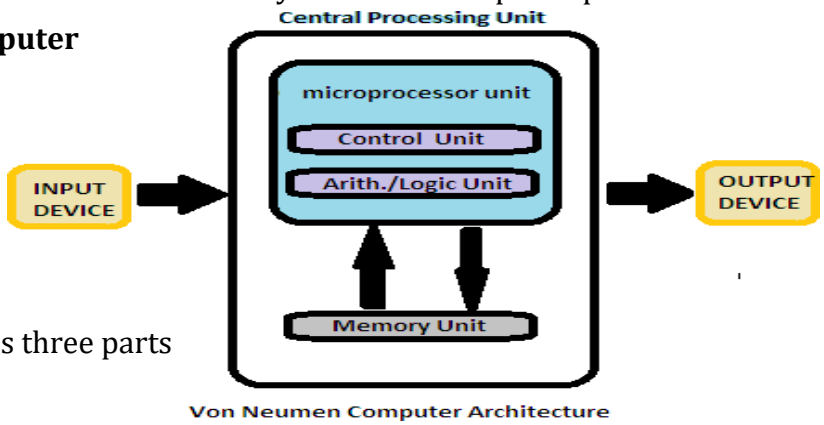
- Data security issue
- Computer crimes
- Health risk
- Bad impact on environment if not properly disposed off

Computer Components Any kind of computers consists of HARDWARE AND SOFTWARE.

Hardware: Computer hardware is the collection of physical elements/parts that constitutes a computer system,such as the monitor, mouse, keyboard, computer data storage, hard drive disk (HDD), system unit (graphic cards, sound cards, memory, motherboard and chips), etc. all of which are physical objects & can be touched.

Software : Software is a generic term for organized collections of computer data and instructions,often broken into two major categories: system software that provides the basic nontask- specific functions of the computer, and application software which is used byusers to accomplish specific tasks.

Functional components of a computer



Von Neumen Computer Architecture

Central processing unit – Comprises three parts

1. Arithmetic/Logic Unit

Performs basic arithmetic operations such as addition and subtraction Performs logical operations such as AND, OR, and NOT. Most modern ALUs have a small amount of special storage units called registers that can be accessed faster than main memory.

2. Control unit

It organizes the computer to work computer as single unit & generates control signals for various devices regarding read/write or execute operation

3. Memory

A collection of cells, each with a u Most computers are byte-addressable

Memory Units – How much memory is required for a file/data/progam etc. is measured by memory units. Following are the memory units.

UNIT	STORAGE
Bit	Binary Digit, Single 1 or 0
Nibble	4 bits
Byte/Octet	8 bits
Kilobyte	1024 bytes
Megabyte	1024 KB
Gigabyte	1024 MB
Terabyte	1024 GB
Petabyte	1024 TB
Exabyte	1024 PB
Zettabyte	1024 EB
Yottabyte	1024 ZB

Memory Types

Primary Memory

Random Access Memory (RAM) - is a type of volatile memory that is stores information on an integrated circuit which hold the data mainly when the program is being executed by the CPU. As it is volatile in nature so it can't store data permanently.

Read Only Memory (ROM) - a non-volatile memory chip in which data are stored permanently, and can not be altered by

Cache Memory - is the volatile computer memory which is very nearest to the CPU,so also called CPU memory, and is between CPU and RAM all the Recent Instructions are Stored into the Cache Memory. It is the fastest memory that provides high-speed data access to a computer microprocessor.

Secondary Storage Devices

A hard disk is a set of stacked disks. Each disk has data recorded electromagnetically in concentric circles, or tracks, on the disk Hard Drive Types

1. Parallel Advanced Technology Attachment (PATA)
2. Serial ATA (SATA)
3. Small Computer System Interface (SCSI)
4. Solid State Drives (SSD)

Upto 12 TB sized HDD is available in the market

Input Devices

Input devices can send data or information to a computer or another device.

Keyboard: It is an input device which sends data in to the computer. The data send depends on the

key pressed by the user.

Mouse: A mouse is a small handheld input device which controls a cursor in a graphical user interface. It can move and select text, files, folders etc. on our computer according to the user input.

Scanner: Scanner optically reads and document, file or image and then changes it into digital signal and sends to the computer.

OMR: optical mark recognition/ reader, is used to read marks on a document and send them tocomputer.

OCR: OCR stands for optical character Recognition, is an input device which reads printed text and sends that to computer.

MICR: Magnetic Ink Character Reader is an input device which generally finds application is banks to process cheques.

Microphone: it receives audio generated by some input source and sends it to a computer. Webcam: it sends the captured images to a computer.

Graphics Tablets: This input device is used to draw using hand.

Trackballs: an upside down mouse ,encased within a socket. Is a cursor control device. Barcode reader: It is used to read the barcode of various items and feed the same to computer. Gamepad: Also known as joy pad is the input controller for video games.

Joystick: these input devices are used to control video games.

Output Devices

A device that can receive data from computer or another device and create output with that data is called output device. Examples of various output devices are as :

Monitor: A monitor is an output device that is responsible for receiving data from a computer and displaying that information as text or images for users to see.

Speakers: Receives sound signal from a computer and then plays that sound signal and thus we hear songs or music or any other audio.

Projector: Gets data from a computer and displays or projects the same information onto a screen or a wall. Projector cannot directly accept data from a user and send that data to another device.

TYPES OF SOFTWARE

Software is an organized instructions/code written by programmers using any of various special computer languages for specific purpose.

Types of software:

(1) System software: controls the basic functions of a computer & hides complexity of computer system from user and application software. E.g. Operating System, Compiler, Interpret etc.

(2) Application software: It handles specialized/ common tasks a user wants to perform, such as banking, hotel management, any data processing, word processing etc.

(3) Utility software: Which helps to manage, maintain and control computer resources. E.g. are antivirus software, backup software

1) **System software**

OPERATING SYSTEM

An Operating System (OS) is a system program that controls and manages the computer resources(resource manager) so that application software can run on it.

Example: Microsoft Windows, Solaris, Linux, MAC OS,Ubuntu, Apple's i-Phone OS etc.

HOW OPERATING SYSTEM WORKS

In any computer or mobile device, the operating system can be termed as the back bone when it comes to software. This is because it has to be there before other programs can be run.It works as a middleman (interface) between machine and user.

At the simplest level, an operating system does two things:

- It manages the hardware resources of the computer system. These resources include such things as the processor, memory, disk space, etc.
- It provides a stable, consistent way for applications to deal with the hardware without having to know all the details of the hardware.

FUNCTIONS OF OPERATING SYSTEM

- Processor management
- Memory management
- Device management
- Storage management
- Application interface
- User interface
- Process management
- Process a program in execution is known as process
- Handling of multiple processes at a time is known as process management.
- Process States

TYPE OF OPERATING SYSTEM

* Single-User, Single Task Operating System:

These operating systems work on single task & single user at a time.E.g. DOS

* Single-User, Multi-Task Operating System:

These operating systems works on more than one task and process them concurrently at a time.E.g. windows 95 or later version of windows

* Multiuser Operating System:

In these OS, multiple users are allowed to access the same data or information at a time via a network. E.g. Unix,Linux,Windows7.

* Multiprocessing Operating System:

Here, a single process runs on two or more processors. All the processing and their management takes place in a parallel way, hence this OS are also called as Parallel Processing. E.g. Linux, UNIX and Windows 7.

* Embedded Operating System:

These are embedded in a device, which is located in ROM.E.g. OS of microwaves,washing machine.

* Distributed Operating System:

In these OS, the computers work in co-operation with each other.

SYSTEM SOFTWARE/PROGRAMMING SOFTWARES

Language processor/Programming tool

As the computer understand machine language(0/1) where as Humans understand High level/Human Lang. Language Processors does the conversion task(high level to machine lang.)

These are of 3 types Language processors

1. Compilers-It convert high-level language code to machine code in one session. It takes time because it have to translate high-level code to lower-level machine language all at once and then save the executable object code to memory.

2. Interpreters-It translates code like a compiler but reads the code and immediately executes that code, and therefore it is faster than a compiler.

3. Assemblers-It translates an assembly language program into machine language. One-pass assemblers go through the source code once. Any symbol used before it is defined will require "errata" at the end of the object telling the linker or the loader to "go back" and overwrite a placeholder which had been left where the as yet undefined symbol was used.

Multi-pass assemblers create a table with all symbols and their values in the first passes, then use the table in later passes to generate code.

(2) Application software

* General Purpose application software

These are ready to use software for daily use purpose

e.g. word processor,spreadsheet,presention,DBMS etc.

* Specific Purpose application software Softwares which are designed for specific task

e.g. Payroll,HotelMgmt,HospitalMgmt,StockMgmt etc.

(3) Utility software/System Utilities

that assist OS in carrying out certain specialized tasks are called utility software.

- Antivirus - An anti-virus scans the system for any virus and if detected, gets rid of it by deleting or isolating it.
- Compression tools - Compression tools are utilities that assist operating systems in shortening files so that they take less space.

(3) Utility software/System Utilities

- Disk Cleanup - Disk cleanup tools assist users in freeing up disk space.
- Disk Defragmenter - Disk defragmenter is a disk management utility that increases file access speeds by rearranging fragmented files on contiguous locations.
- Backup - Backup utility enables backing up of files, folders, databases or complete disks.
- File management tools - Utility software providing regular file management tasks like browse, search, update, preview, etc. are called file management tools.
- Restore – This utility restores the backup earlier taken.
- Device driver or hardware driver is a group of files that enable one or more hardware devices to communicate with the computer's operating system. Without drivers, the computer would not be able to send and receive data correctly to hardware devices, such as a printer

MCQ

1. Smallest measurement unit of computer memory is?
(a) Megabyte
(b) Bit
(c) Byte
(d) Killo Byte

2. How many bytes are in 1 Kilobyte?
(a) 8 Bytes
(b) 128 Bytes
(c) 1024 Bytes
(d) 256 Bytes

3. Read Only Memory (ROM) is a_____memory.
(a) Non Volatile Memory
(b) Volatile Memory
(c) Both (a & b)
(d) None of these

4. Which of the following is designed to control the operations of a computer?
a) Application Software
b) System Software
c) Utility Software
d) User

5. The software designed to perform a specific task:
a) Synchronous Software
b) Package Software
c) Application Software
d) System Software

6. The software substituted for hardware and stored in ROM.
a) Synchronous Software
b) Package Software
c) Firmware
d) Middleware

7. Which of the following is not application software?
a) Windows 7
b) WordPad
c) Photoshop
d) MS-excel

Very Short Answer Questions

1. Namethesoftwarerequiredtomakeacomputerfunctional.Writedownitstwoprimaryservices
.
2. What is the need for secondary memory?
3. Draw the block diagram of a computer system. Briefly write about the functionality of each component.

Short Answer Questions

1. State the basic units of Computer along with its sub units and their functions.
2. Differentiate between RAM and ROM.
3. What is the role of CPU in Computer System?

Long Answer Questions

1. Name any four secondary storage media.
2. Define software. Explain with examples- System Software, Utility Software and Application Software.
3. Write short notes on Assembler, Compiler and Interpreter.

Boolean Logic

Boolean Logic

Because of computer understands machine language(0/1) which is binary value so every operation is done with the help of these binary value by the computer.

To understand boolean logic properly we have to understand Boolean logic rule,Truth table and logic gates

Boolean Logic rules

Boolean Algebra is the mathematics we use to analyse digital gates and circuits. We can use these "Laws of Boolean" to both reduce and simplify a complex Boolean expression in an attempt to reduce the number of logic gates required.

A +1=1	Annulment
A +0=A	Identity
A .1=A	Identity
A.0 =0	Annulment
A +A=A	Idempotent
A .A=A	Idempotent
NOTA=A	Double Negation
A +A =1	Complement
A .A =0	Complement
A+B=B+A	Commutative
A.B =B.A	Commutative
A+B =A.B	deMorgan'sTheorem
A.B=A+B	deMorgan'sTheorem

Boolean Expression

A Boolean expression is a logical statement that is either TRUE or FALSE .

A Boolean expression can consist of Boolean data, such as the following:

* BOOLEAN values (YES and NO, and their synonyms, ON and OFF, and TRUE and FALSE)
* BOOLEAN variables or formulas
* Functions that yield BOOLEAN results

De Morgan's Law

The complement of the union of two sets is equal to the intersection of their complements and the complement of the intersection of two sets is equal to the union of their complements. These are called De Morgan's laws.

For any two finite sets A and B

(i) (A+B)'=A'.B'

(ii) (A . B)'=A'+B'

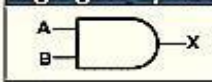
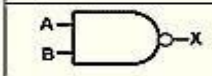
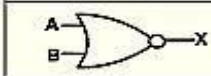
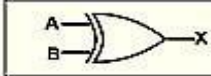
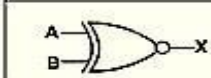
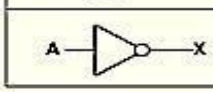
Truth table

A truth table is a mathematical table used in logic. e.g.

A	B	(A and B)	(A or B)	not(A and B)	not(A or B)
True	True	True	True	False	False
True	False	False	True	True	False
False	True	False	True	True	False
False	False	False	False	True	True

Logic Gates

Logic gate is an idealized or physical device implementing a Boolean function. These are used to construct logic circuit.

LOGIC GATES		
Logic gate symbol	Description	Boolean
	The AND gate output is at logic 1 when, and only when all its inputs are at logic 1, otherwise the output is at logic 0.	$X = A \cdot B$
	The OR gate output is at logic 1 when one or more of its inputs are at logic 1. If all the inputs are at logic 0, the output is at logic 0.	$X = A + B$
	The NAND Gate output is at logic 0 when, and only when all its inputs are at logic 1, otherwise the output is at logic 1.	$X = \overline{A \cdot B}$
	The NOR gate output is at logic 0 when one or more of its inputs are at logic 1. If all the inputs are at logic 0, the output is at logic 1.	$X = \overline{A + B}$
	The XOR gate output is at logic 1 when one and ONLY ONE of its inputs is at logic 1. Otherwise the output is logic 0.	$X = A \oplus B$
	The XNOR gate output is at logic 0 when one and ONLY ONE of its inputs is at logic 1. Otherwise the output is logic 1. (It is similar to the XOR gate, but its output is inverted).	$X = \overline{A \oplus B}$
	The NOT gate output is at logic 0 when its only input is at logic 1, and at logic 1 when its only input is at logic 0. For this reason it is often called an INVERTER.	$X = \overline{A}$

Universal gates are the logic gates which are capable of implementing any Boolean function without requiring any other type of gate.

Types of Universal Gates-

In digital electronics, there are only two universal gates which are-

1. NAND Gate
2. NOR Gate

Number System & Encoding Schemes

In general term computer represent information in different types of data forms i.e. number , character ,picture ,audio , video etc.

Computers are made of a series of switches/ gates. Each switch has two states: ON(1) or OFF(0).That's why computer works on the basis of binary number system(0/1).But for different purpose different number systems are used in computer world to represent information. E.g. Octal, Decimal, Hexadecimal.

NUMBER SYSTEM		
SYSTEM	BASE	DIGIT
Binary	2	0 1
Octal	8	0 1 2 3 4 5 6 7
Decimal	10	0 1 2 3 4 5 6 7 8 9
Hexadecimal	16	0 1 2 3 4 5 6 7 8 9 A B C D E F

CONVERSIONS

DECIMAL TO OTHER

1. DECIMAL TO BINARY

Decimal Number System to Other Base

To convert Number system from **Decimal Number System** to **Any Other Base** is quite easy;you have to follow just two steps:

A) Divide the Number(Decimal Number)by the base of target base system(in which you want to convert the number:Binary (2),octal (8)and Hexadecimal(16)).

B) Write the remainder from step1 as a Least Signification Bit(LSB)to Step last as a Most significant Bit(MSB).

Decimal to Binary Conversion	Result
Decimal Number is : **(12345)10** 2 | 12345 | 1 LSB 2 | 6172 | 0 2 | 3086 | 0 2 | 1543 | 1 2 | 771 | 1 2 | 385 | 1 2 | 192 | 0 2 | 96 | 0 2 | 48 | 0 2 | 24 | 0 2 | 12 | 0 2 | 6 | 0 2 | 3 | 1 1 | 1 MSB	BinaryNumberis **(11000000111001)2**

2. DECIMAL TO OCTAL

Decimal to Octal Conversion	Result
Decimal Number is:**(12345)10** 8 | 12345 — 1 LSB 8 | 1543 — 7 8 | 192 — 0 8 | 24 — 0 3 — 3 MSB	Octal Number is **(30071)8**

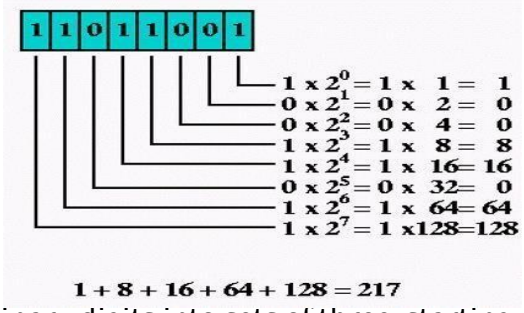
3. DECIMAL TO HEXADECIMAL

Decimal to Hexadecimal Conversion	Result
Example1 Decimal Number is:**(12345)10** 16 | 12345 — 9 LSB 16 | 771 — 3 16 | 48 — 0 8 | 3 — 3 MSB	Hexadecimal Number is**(3039)16**
Example2 Decimal Number is:**(725)10** 16 | 725 — 5 LSB 16 | 45 — 13 (D) 2 — 2 MSB	Hexadecimal Number is **(2D5)16** **Convert** **10,11,12,13,14,15** **To its equivalent…A,B,C,D,E,F**

BINARY TO OTHER

A)Multiply thedigit with 2(with place value exponent).Eventually add all the multiplication becomes the Decimal number.

1. BINARY TODECIMAL



2. BINARY TO OCTAL

An easy way to convert from binary to octal is to group binary digits into sets of three, starting with the least significant (rightmost)digits.

Binary:11100101 =	11 100 101	
	011 100 101	Pad the most significant digits with zeros if necessary to complete a group of three.

Then,lookup each group in a table:

Binary:	000	001	010	011	100	101	110	111

Octal:	0	1	2	3	4	5	6	7

Binary=	011	100	101	
Octal =	3	4	5	= 345oct

3. BINARYTOHEXADECIMAL

An equally easy way to convert from binary to hexadecimal is to group binary digits into sets off our,starting with the least significant (rightmost)digits.

Binary:11100101 =11100101

Then, lookup each group in a table:

Binary:	0000	0001	0010	0011	0100	0101	0110	0111
Hexadecimal:	0	1	2	3	4	5	6	7
Binary:	1000	1001	1010	1011	1100	1101	1110	1111
Hexadecimal:	8	9	A	B	C	D	E	F

Binary=	1110	0101	
Hexadecimal=	E	5	=E5 hex

OCTAL TO OTHER

1. OCTAL TO BINARY

Converting from octal to binary is as easy as converting from binary to octal. Simply lookup each octal digit to obtain the equivalent group of three binary digits.

Octal:	0	1	2	3	4	5	6	7
Binary:	000	001	010	011	100	101	110	111

Octal =	3	4	5	
Binary=	011	100	101	=011100101binary

2. OCTALTOHEXADECIMAL

When converting from octal to hexadecimal, it is often easier to first convert the octal number into binary and then from binary into hexadecimal. For example, to convert 345octal into hex:

(from the previous example)

Octal =	3	4	5	
Binary=	011	100	101	=011100101binary

Drop any leading zeros or pad with leading zeros to get groups of four binary digits (bits):Binary 011100101=11100101

3. OCTALTODECIMAL

The conversion can also be performed in the conventional mathematical way,by showing each digit place as an increasing power of 8.

345 octal=$(3*8^2) +(4*8^1) +(5*8^0) =(3*64) +(4*8) + (5*1)=229$decimal

HEXADECIMAL TO OTHER

1. HEXADECIMALTO BINARY

Converting from hexadecimal to binary is as easy as converting from binary to hexadecimal.Simply lookup each hexadecimal digit to obtain the equivalent group off our binary digits.

Hexadecimal:	0	1	2	3	4	5	6	7
Binary:	0000	0001	0010	0011	0100	0101	0110	0111
Hexadecimal:	8	9	A	B	C	D	E	F
Binary:	1000	1001	1010	1011	1100	1101	1110	1111

Hexadecimal	A	2	D	E	
Binary=	1010	0010	1101	1110	10100010110 binary

2. HEXADECIMAL TO DECIMAL

Convert 42A.1216 into a decimal number. Solution-

The hexadecimal number given is 42 A.12

Positional weights 210-1-2

The positional weights for each of the digits are written in italics below each digit. Hence the decimal equivalent number is given as:

$4\times16^{2}+2\times16^{1}+10\times16^{1}+1\times16^{-1}+1\times16^{-2}$

=1024 +32 +10+0.0625 +0.00390625

$=(1066.06640625)_{10}$

3. HEXADECIMALTOOCTAL

Given hexa decimalnumber is A 72E

Binary equivalent is 1010011100101110=1010011100101110

Forming groups of 3bits from the LSB 001 010011100101 110

Octal equivalent 1 23 45 6

Hence the octal equivalent of $(A72E)_{16}$is $(123456)_{8}$.

ENCODING SCHEMES

American Standard Code for Information Interchange (ASCII)

In the early 1960s, computers had no way of communicating with each other due to different ways of representing keys of the keyboard. Hence, the need for a common standard was realised to overcome this shortcoming. Thus, encoding scheme ASCII was developed for standardising the character representation. ASCII is still the most commonly used coding scheme.

Initially ASCII used7 bit store present characters. Recall that there are only binary digits(0or1).Therefore, total number of different character son the English keyboard that can be encoded by 7-bit ASCII code is 27 = 128. Following Table shows some printable characters for ASCII code. But ASCII is able to encode character set of English language only.

Indian Script Code for Information Interchange(ISCII)

In order to facilitate the use of Indian languages on computers, a common standard for coding Indian scripts called ISCII was developed in India during mid1980s.

It is an 8-bit code representation for Indian languages which means it can represent 28=256characters.It retainsall 128 ASCII codes and uses rest of the codes(128)for additional Indian language character set. Additional codes have been assigned in the upper region (160–255)for the 'aksharas' of the language.

UNICODE

There were many encoding schemes, for character sets of different languages. But they were not able to communicate with each other, as each of them represented characters in their own ways. Therefore, a standard called UNICODE has been developed to incorporate all the characters of every written language of the world. Commonly used UNICODE encodings are UTF-8, UTF-16 and UTF-32.

MCQ

1. When we convert 10011 binary numbers to decimals. Then the solution is :
   a. 20 b.18 c.19 d.16

2. Convert (22) from octal to its corresponding decimal equivalent.
   a. 20 b.18 c.14 d. 81

3. Name the number system which uses alphabets as well as numerical.
   a. Binary number system
   b. octal number system
   c. Decimal number system
   d. Hexadecimal number system

4. The octal equivalent of $(13)_{10}$ is
   a. 18 b. 14 c. 15 d. 16

5. Conversion of hexadecimal number $(69)_{16}$ to octal equivalent will be
   a.451 b. 351 c. 251 d. 151

Very Short Answer Questions

1. Write full form of ASCII and ISCII.
2. What is the base of binary number system?
3. How many digits used by the hexadecimal number system.

Short Answer Questions

1. Expressthefollowingoctalnumbersintotheirequivalentdecimalnumbers.
   (i) 145 (ii) 6760 (iii)455 (iv)10.75
2. Convert the following binary numbers into octal and hexadecimal numbers.

(i)1110001000 (ii)110110101(iii)1010100 (iv)1010.1001

UNIT -2

Getting Stating with Python

1. INTRODUCTION

- Computers are used for solving various day-to-day problems.
- It is pertinent to mention that computers themselves can not solve a problem.
- Precisestep-by-stepinstructionsshouldbegivenbyustosolvetheproblem.
- Thus, the success of a computer in solving a problem depends on how correctly and precisely we define the problem, design a solution (algorithm) and implement the solution (program) using a programming language.
- Thus, problem solving is the process of identifying a problem, developing an algorithm for the identified problem and finally implementing the algorithm to develop a computer program.

2. Steps for ProblemSolving

There are four steps in problem solving

- Analysing the problem, Developing an Algorithm, Coding, Testing and Debugging

2.1 Analysing the problem

- It is important to clearly understand a problem before we begin to find the solution for it.
- If we are not clear as to what is to be solved,we may end up developing a program which may not solve our purpose.
- By analysing a problem, we would be able to figure out what are
- the inputs that our program should accept and the outputs that it should produce.

2.2 Developing an Algorithm

- The solution for a problem is represented in step by step procedure called analgorithm.
- For a given problem, more than one algorithm is possible and we have to select the most suitable solution.

2.3 Coding

- After finalising the algorithm, we need to convert the algorithm into the format which can be understood by the computer to generate the desired solution.

2.4 Testing and Debugging

- The program created should be tested on various parameters.
- The program should meet the requirements of the user.
- In the presence of syntactical errors, no output will be obtained.
- In case the output generated is in correct, then the program should be checked for logical errors, if any.

3. Algorithm

Algorithm is the step by step procedure for solving the problem. Suppose following are the steps required for an activity 'riding a bicycle':

- Remove the bicycle from the stand,
- Sit on the seat of the bicycle,
- Start peddling,
- Use breaks when ever needed and
- Stop on reaching the destination.

Example:

Algorithm to find square of anumber.

Step1: Input a number and store it to num

Step 2: Compute num * num and store
it in square

Step 3:Print square

3.1 Why do we need an Algorithm

- Writing an algorithm is mostly considered as a first step to programming.
- Once we have an algorithm to solve a problem, we can write the computer program for giving instructions to the computer in high level language.
- If the algorithm is correct, computer will run the program correctly, everytime.
- So, the purpose of using an algorithm is to increase the reliability, accuracy and efficiency of obtaining solutions.

Characteristics of a good algorithm

• Precision—the steps are precisely stated or defined.

• Uniqueness—results of each step are uniquely defined and only

depend on the input and the result of the preceding steps.

- Finiteness—the algorithm always stops after a finite number of steps.
- Input—the algorithm receives some input.
- Output—the algorithm produces some output.

While writing an algorithm ,it is required to clearly identify the following:

- The input to be taken from the user
- Processing or computation to be performed to get the desired result
- The output desired by the user

4. Representation of Algorithms

There are two common methods of representing an algorithm —flowchart and pseudocode. Either of the methods can be used to represent an algorithm while keeping in mind the following:

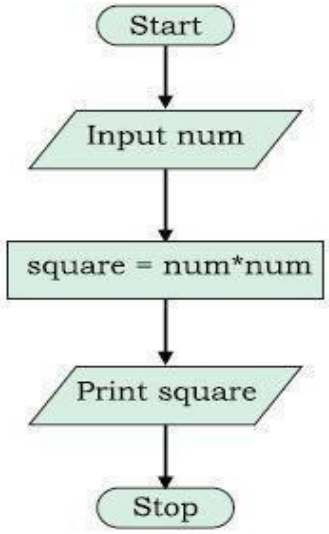
- it show cases the logic of the problem solution, excluding any implementational details
- it clearly reveals the flow of control during execution of the program

4.1 Flowchart —Visual Representation of Algorithms

- A flow chart is a visual representation of an algorithm.
- A flowchart is a diagram made up of boxes, diamonds and other shapes, connected by arrows.

Symbol	Name	Function
	Start/end	An oval represents a start or end point
	Arrows	A line is a connector that shows relationships between the representative shapes
	Input/Output	A parallelogram represents input or output
	Process	A rectagle represents a process
	Decision	A diamond indicates a decision

Flow chart to calculate square of a number



4.2 Pseudocode

A pseudocode (pronounced Soo-doh-kohd) is another way of representing an algorithm. It is considered as a non-formal language that helps programmers to write algorithm. The word"pseudo" means "not real," so "pseudocode" means "not real code". Following are some of the frequently used keywords while writing pseudocode:

- INPUT/•COMPUTE/•PRINT/•INCREMENT/•DECREMENT
- IF/ELSE,•WHILE,•TRUE/FALSE

Example:

Pseudo code for the sum of two numbers will be:

Input num1
input num2
COMPUTE Result = num1 + num2
PRINT Result

4.3 Coding

- Once analgorithm is finalised,it should be coded in ahigh-level programming language as selected by the programmer.
- The ordered set of instructions are written in that programming language by following its syntax.
- Syntax is the set of rules or grammar that governs the formulation of the statements in the language,such as spellings,order of words, punctuation,etc.

4.4 Decomposition

- The basic idea of solving a complex problem by decomposition is to 'decompose' or break down a complex problem into smaller sub problems.

Answer the Following Questions (Very ShortAnnswers)

1. Define Algorithm
2. What is decomposition?
3. Why do we need Algorithm?
4. What is meant by Debugging?

Answer the Following Questions (Short Answers)

1. Write an algorithm to find the greatest among two different numbers.
2. Write a pseudo code to calculate the factorial of a number.
3. Write an algorithm to find greater among three numbers

Answer the Following Questions (Long Answers)

1. Write pseudo code and draw flow chart to accept number still the user enters and then find their average.
2. Write a pseudocode and draw a flowchart where multiple conditions are checked to categorize a person as either child(<13), teenager(>=13but<20) or adult(>=20),based on age specified:

3. Write an algorithmth at accepts four numbers as input and find the largest and smallest of them.

PYTHON PROGRAMMING FUNDAMENTAL:

5.1.1 Features of Python

• Python is a high level language. It is a free and open source language.
• It is an interpreted language, as Python programs are executed by an interpreter.
• Python programs are easy to understand as they have a clearly defined syntax and relatively simple structure.
• Python is case-sensitive. For example, NUMBER and number are not same in Python.
• Python is portable and platform independent, means it can run on various operating systems and hardware platforms.
• Python has a rich library of predefined functions.
• Python uses indentation for blocks and nested blocks.

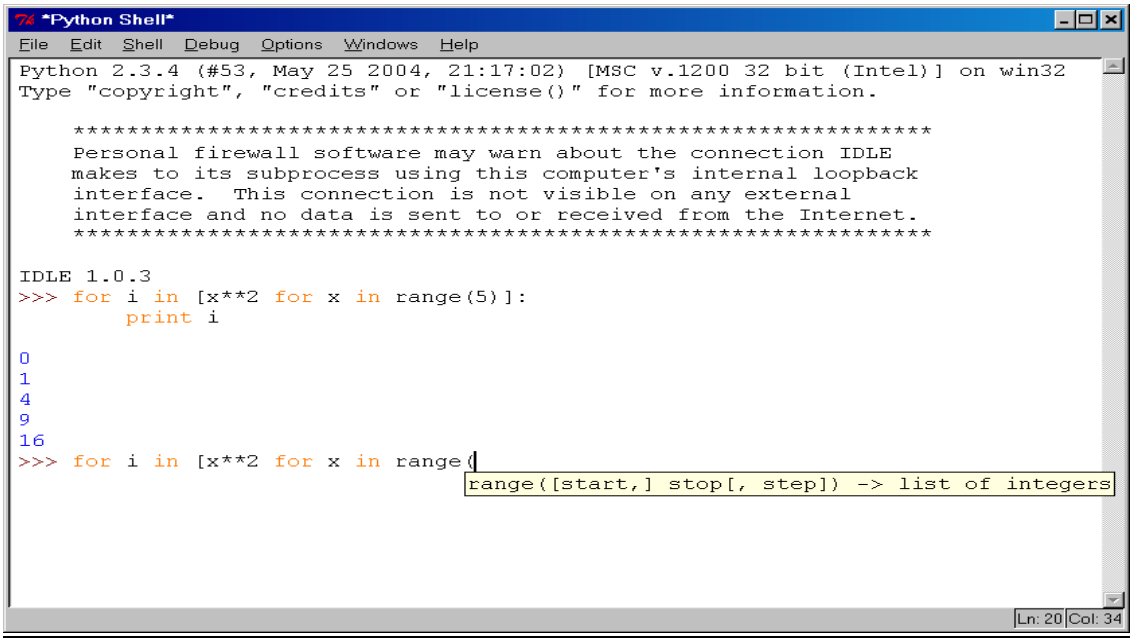
How to Install Python

- Python is pre-installed on most Unix systems, including Linux and MAC OS X
- The pre-installed version may not be the most recent one (2.6.2 and 3.1.1 as of Sept 09)
- Download from http://python.org/download/
- Python comes with a large library of standard modules
- There are several options for an IDE

IDLE – works well with Windows

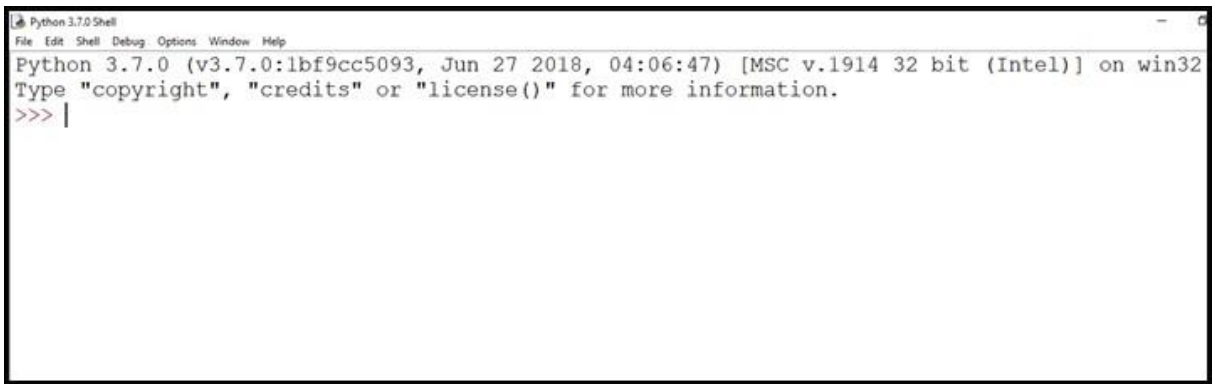
IDLE (Integrated Development Environment)

- IDLE is an Integrated DeveLopment Environment for Python, typically used on Windows
- Multi-window text editor with syntax highlighting, auto-completion, smart indent and other.
- Python shell with syntax highlighting.
- Integrated debugger with stepping, persistent breakpoints, and call stack visibility



5.1.2 Working with Python

To write and run (execute) a Python program, we need to have a Python interpreter installed on our computer or we can use any online Python interpreter. The interpreter is also called Python shell. A sample screen of Python interpreter is shown in Figure:



In the above screen, the symbol >>> is the Python prompt, which indicates that the interpreter is ready to take instructions. We can type commands or statements on this prompt to execute them using a Python interpreter.

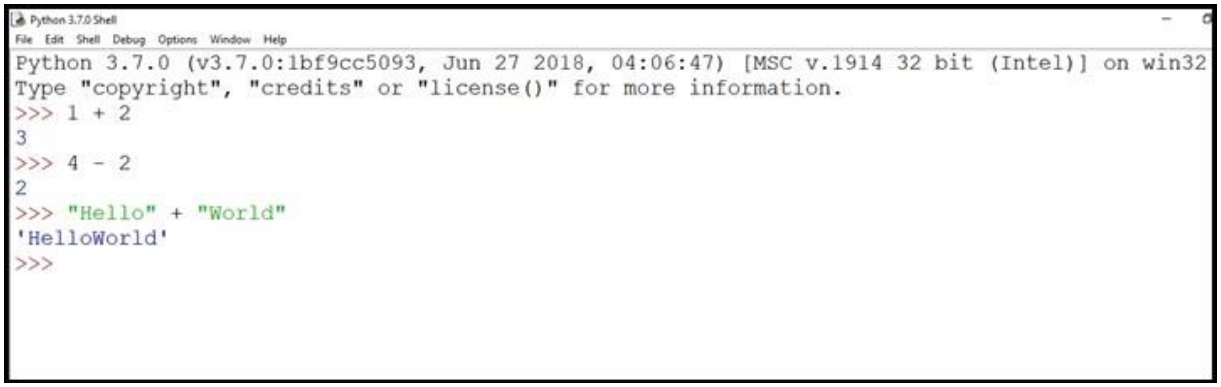
5.1.3 Execution Modes

There are two ways to use the Python interpreter:
a) Interactive mode
b) Script mode

Interactive mode allows execution of individual statement instantaneously. Whereas, Script mode allows us to write more than one instruction in a file called Python source code file that can be executed.

(A) Interactive Mode

To work in the interactive mode, we can simply type a Python statement on the >>> prompt directly. As soon as we press enter, the interpreter executes the statement and displays the result(s), as shown in Figure:



Working in the interactive mode is convenient for testing a single line code for instant execution. But in the interactive mode, we cannot save the statements for future use and we have to retype the statements to run them again.

(B) Script Mode

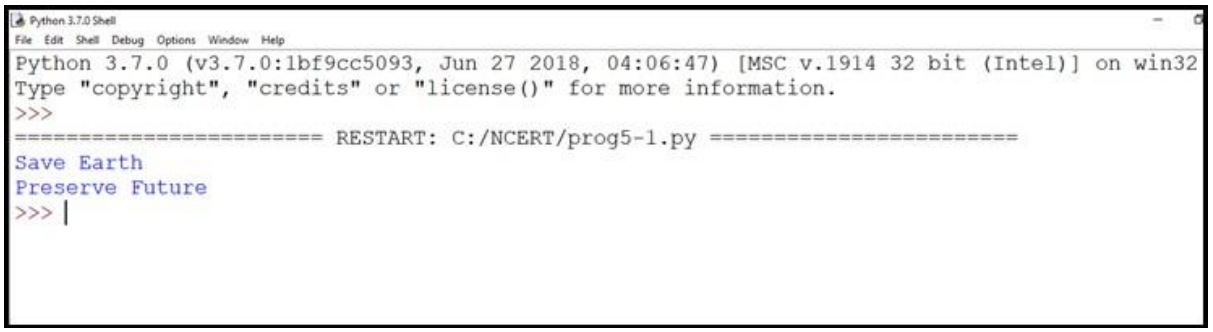
In the script mode, we can write a Python program in a file, save it and then use the interpreter to execute it. Python scripts are saved as files where file name has extension ".py". By default, the Python scripts are saved in the Python installation folder. To execute a script, we can either:

a) Type the file name along with the path at the prompt. For example, if the name of the file is prog5-1.py, we type prog5-1.py. We can otherwise open the program directly from IDLE.

b) While working in the script mode, after saving the file, click [Run]->[Run Module] from the menu as shown in Figure below.

c) The output appears on shell as shown in figure below:

Program 5-1 Write a program to show print statement in script mode.



Basic DataTypes:

- **Integers (default for numbers)- Number without decimal point**
  z = 5 / 2 # Answer 2.5, integer division
- **Floats : Number with decimal Point**
  x = 3.456

- **Strings -**

Can use "" or '' to specify with "abc" == 'abc'

Use triple double-quotes for multi-line strings or strings than contain both ' and " inside of them:

"""a'b"c"""

Comments :

- **Comments are very important while writing a program. It describes what's going on inside a program so that a person looking at the source code does not have a hard time figuring it out.**
- **In Python, we use the hash (#) symbol to start writing a comment.**
- **If we have comments that extend multiple lines, one way of doing it is to use hash (#) in the beginning of each line.**
- **For example:**

```
#This is a long comment
#and it extends
#to multiple lines
```

- **Another way of doing this is to use triple quotes, either ''' or """.**
- **These triple quotes are generally used for multi-line strings. But they can be used as multi-line comment as well.**

```
"""This is also a
perfect example of
multi-line comments"""
```

Assignment :

- ***Binding a variable*** **in Python means setting a *name* to hold a *reference* to some *object***
  Assignment creates references, not copies
- **Names in Python do not have an intrinsic type, objects have types**
  Python determines the type of the reference automatically based on what data is assigned to it
- **You create a name the first time it appears on the left side of an assignment expression: x = 3**
- **A reference is deleted via garbage collection after any names bound to it have passed out of scope**
- **Python uses *reference semantics* (more later)**
- **You can assign to multiple names at the same time**

```
>>> x, y = 2, 3
>>> x
2
>>> y
3
```

- **This makes it easy to swap values**

```
>>> x, y = y, x
```

- **Assignments can be chained**

```
>>> a = b = x = 2
```

Python keywords

Reserved words in the library of a language.There are 3 3keywords in python.

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

All the keywords are in lowercase except 03 keywords(True, False, None)

Identifier

- **An identifier is a name given to entities like class, functions, variables, etc.**
- **It helps to differentiate one entity from another.**

Rules for naming Identifier :

- **Identifiers can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore _.**
- **Names like myClass, var_1 and print_this_to_screen, all are valid example.**
- **An identifier cannot start with a digit. 1variable is invalid, but variable1 is perfectly fine.**
- **Keywords cannot be used as identifiers.**
- **We cannot use special symbols like !, @, #, $, % etc. in our identifier.**

bob Bob _bob _2_bob_ bob_2 BoB

- **There are some reserved words:**

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

Program : Write a program to display values of variables in Python.

```
#To display value of variable
message = "hello"
print(message)
)
```

Output:
Hello

Program : Write a Python program to find the area of a rectangle given that its length is 10 units and breadth is 20 units.

```
#To find the area of a rectangle
length = 10
breadth = 20
area = length * breadth
print("Area of Rectangle=", area)
```

Output: 200

Do it yourself:

1. Write a Python program to find the sum and subtract of two numbers
2. Find the simple interest. Principle amount, rate of interest and time given by user.

Constants

- A constant is a type of variable whose value cannot be changed.
- It is helpful to think of constants as containers that hold information which cannot be changed later.

Literals

- Literals
- Literal is a raw data given in a variable or constant. In Python, there are various types of literals they are as follows:
    - Numeric Literals
    - Numeric Literals are immutable (unchangeable). Numeric literals can belong to 3 different numerical types Integer, Float and Complex.
    - a = 0b1010 #Binary Literals
    - b = 100 #Decimal Literal
    - c = 0o310 #Octal Literal
    - d = 0x12c #Hexadecimal Literal
- #Float Literal
    - float_1 = 10.5
    - float_2 = 1.5e2
- #Complex Literal
    - x = 3.14j
- print(a, b, c, d)
- print(float_1, float_2)
- print(x, x.imag, x.real)

OUTPUT

10 100 200 300
10.5 150.0
3.14j 3.14 0.0

Python Data Types

Built-in Data Types

In programming, data type is an important concept. Variables can store data of different types, and different types can do different things. Python has the following data types built-in by default, in these categories:

	Str
Numeric Types:	int, float, complex
Sequence Types:	list, tuple, range
Mapping Type:	dict
Set Types:	set, frozen set
Boolean Type:	bool

Binary Types: bytes, byte array, memory view

Let us now try to execute few statements in interactive mode to determine the data type of the variable using built-in function type().
Example :

```
>>> num1 = 10
>>> type(num1)
<class 'int'>
>>> num2 = -1210
>>> type(num2)
<class 'int'>
>>> var1 = True
>>> type(var1)
<class 'bool'>
```

Operators

An operator is used to perform specific mathematical or logical operation on values. The values that the operators work on are called operands. For example, in the expression 10 + num, the value 10, and the variable num are operands and the + (plus) sign is an
operator. Python supports several kinds of operators whose categorisation is briefly explained in this section

Python divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators

Python Arithmetic Operators

Arithmetic operators are used with numeric values to perform common mathematical operations:

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Concatenation

Concatenating means obtaining a new string that contains both of the original strings. In Python, there are a few ways to concatenate or combine strings. The new string that is created is referred to as a string object. In order to merge two strings into a single object, you may use the + operator.

For Example
>>>"CBSE "+"India"

Output is CBSEIndia

Replication. : The multiplication operator acts as a replication operator when we have one string and one integer as operands. What is replication? First, understand the meaning of the word replication**.**

>>>"CBSE" 3
Output is CBSECBSECBSE

Python Assignment Operators		
Operator	**Example**	**Equal to**
=	a = 20	a = 20
+=	a += b	a = a + b
-=	a -= b	a = a - b
*=	a *= b	a = a * b
/=	a /= b	a = a / b
%=	a %= b	a = a % b
//=	a //= b	a = a // b
**=	a **= b	a = a ** b
&=	a &= b	a = a & b
|=	a |= b	a = a | b
^=	a ^= b	a = a ^ b
>>=	a >>= b	a = a >> b
<<=	a <<= b	a = a << b

Python Comparison Operators

Comparison operators are used to compare two values:

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Python Logical Operators

Logical operators are used to combine conditional statements:

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Python Identity Operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Python Membership Operators

Membership operators are used to test if a sequence is presented in an object:

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

The following table lists all operators from highest precedence to lowest.

Operator	Description
**	Exponentiation (raise to the power)
~ + -	Complement, unary plus and minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'td>
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

Statement

In Python, a statement is a unit of code that the Python interpreter can execute.

Example

```
>>> x = 4        #assignment statement
>>> cube = x ** 3       #assignment statement
>>> print (x, cube)      #print statement

        4 64
```

- **Multi-line statement**
  In Python, end of a statement is marked by a newline character. But we can make a statement extend over multiple lines with the line continuation character (\).

For example:

```
a = 1 + 2 + 3 + \
4 + 5 + 6 + \
7 + 8 + 9
```

Input and Output

Sometimes, a program needs to interact with the user's to get some input data or information from the end user and process it to give the desired output. In Python, we
have the input() function for taking the user input. The input() function prompts the user to enter data. It accepts all user input as string. The user may enter a number or a string but the input() function treats them as strings only. The syntax for input() is:

input ([Prompt])

Example

```
>>>fname = input("Enter your first name: ")
Enter your first name: Arnab

>>> age = input("Enter your age: ")
Enter your age: 19

>>> type(age)
<class 'str'>
```

Example

```
#function int() to convert string to integer
>>> age = int( input("Enter your age:"))

Enter your age: 19
>>> type(age)
<class 'int'>
```

Example

Statement	**Output**
print("Hello")	Hello
print(10*2.5)	25.0
print("I" + "love" + "my" + "country")	Ilovemycountry
print("I'm", 16, "years old")	I'm 16 years old

More Examples:

```
print(1,2,3,4)
# Output: 1 2 3 4

print(1,2,3,4,sep='*')
# Output: 1*2*3*4

print(1,2,3,4,sep='#',end='&')
# Output: 1#2#3#4&
```